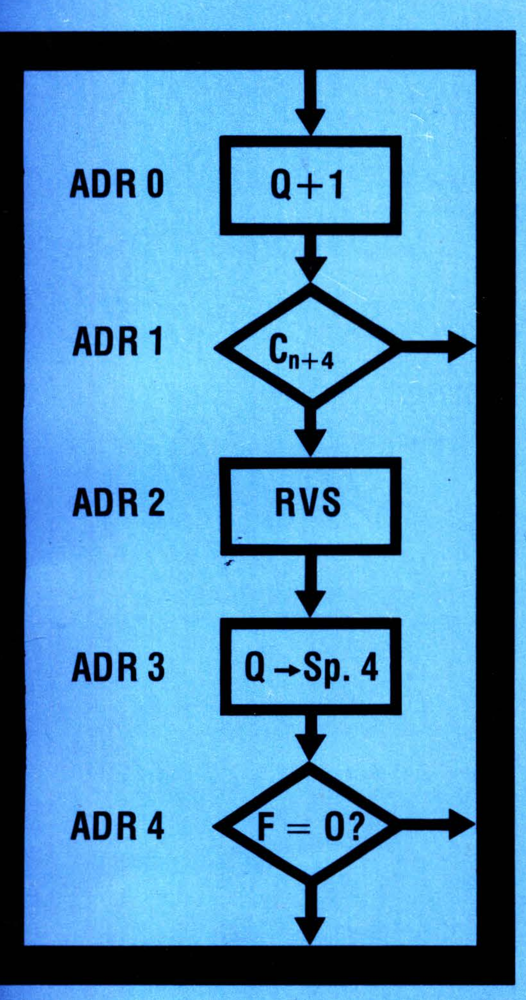




Buchreihe Elektronik **161**

Herbert Bernstein

TTL Mikroprozessor System


TOPP
Buchreihe Elektronik 161

Herbert Bernstein

TTL-Mikroprozessor-System

Aufbau, Bauanleitung
und Programmierung
eines Mikrocomputers

Frech-Verlag Stuttgart

ISBN 3-7724-0308-5

Satz: Fotosatz Hauer, Stuttgart
Druck: Frech, Stuttgart

Inhaltsverzeichnis

Vorwort

Seit der „Elektronica“ von 1976 in München ist jedem Elektroniker der Begriff des Mikroprozessors bekannt. Bei vielen Elektronikern besteht seitdem der Wunsch nach einem selbstgebauten System, das er in- und auswendig kennt. Dieses System soll aber die Möglichkeiten bieten, auf jeden anderen Mikroprozessor schnell umsteigen zu können.
Mit diesem Buch ist selbst ein technisch interessierter Laie in der Lage, sich einen preiswerten Heimcomputer mit einem Mikroprozessor aufzubauen und zu programmieren.

Als Basis dient der von der Firma AMD (Advanced Micro Devices) entwickelte Mikroprozessor 2901. Folgende Firmen stellen diesen Typ als Zweitlieferanten her:

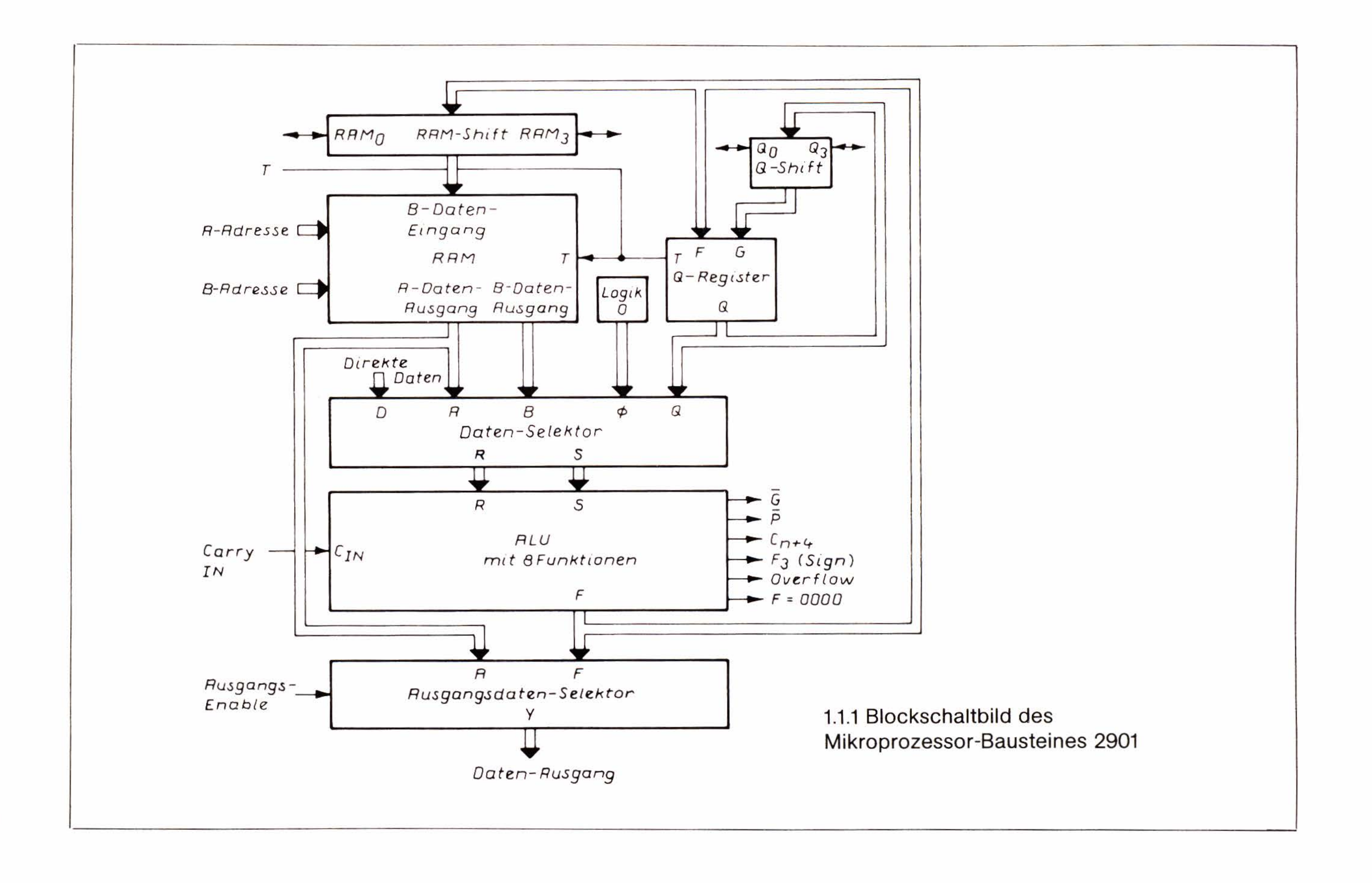
AMD (Erstlieferant)
Fairchild
Monolithic Memories
Motorola
National Semiconductor
Raytheon
Sescosem
Siemens
Signetics.

Neben einer ausführlichen Bauanleitung und Beschreibung der einzelnen Bausteine finden Sie 22 Programmierungsbeispiele. Die Bauelemente erhalten Sie in jedem gut sortierten Elektronik-Fachgeschäft.
An dieser Stelle möchte sich der Autor für die großzügige Unterstützung der Firma COSMOS bedanken.

1. Bausteine für das Mikrocomputer-System

1.1 Mikroprozessor 2901

Der Mikroprozessor 2901 ist ein schneller TTL-Baustein. Er bildet mit seinen zahlreichen Funktionen das Gehirn des Mikrocomputer-Systems.
In Bild 1.1.1 ist die Funktion des Bausteines 2901 gezeigt. Kernstück bildet die 8-Funktionen-ALU. Eine ALU ist die arithmetische und logische Einheit (Unit). Aus den beiden Eingängen R und S werden die einzelnen Funktionen gebildet. Die Funktionen erhalten wir an dem Ausgang F. Die ALU hat einen CARRY-IN-Eingang für die Überträge. Die Ausgänge $\overline{G}$ und $\overline{P}$ dienen zur Erweiterung, werden aber bei diesem System nicht benötigt. Der Ausgang C_{n+4} ist der Übertrags-Ausgang. Erscheint bei einer Rechenoperation ein Übertrag, erhält dieser Ausgang ein Signal. Der Ausgang F_3 ist der Sign-Ausgang, also das Vorzeichen. Anhand des Signals erkennen wir, ob das Ergebnis positiv oder negativ ist. Der Ausgang OVERFLOW kennzeichnet uns, ob ein Überlauf in der Rechenoperation entstanden ist. Der Ausgang F zeigt uns an, ob das Ergebnis ein 0-Wort ist. Die ALU erhält die Daten aus dem Daten-Selektor. Der Daten-Selektor hat fünf Eingänge und erzeugt aus diesen die beiden Dateneingänge R und S für die ALU. An dem Daten-Selektor liegen die direkten Eingangsdaten D, die gespeicherten Daten von dem A-RAM und von dem B-RAM, die Ø-Logikdaten (ein 0-Wort) und die Daten des Q-Registers. Durch Befehle können wir die fünf Eingänge entsprechend auf die ALU schalten. Damit ergeben sich mehrere Variationsmöglichkeiten. Die direkten Daten können mit den anderen vier Eingängen verbunden werden. Welche Verbindung gewählt wird, ist von dem jeweiligen Anwendungsfall abhängig.



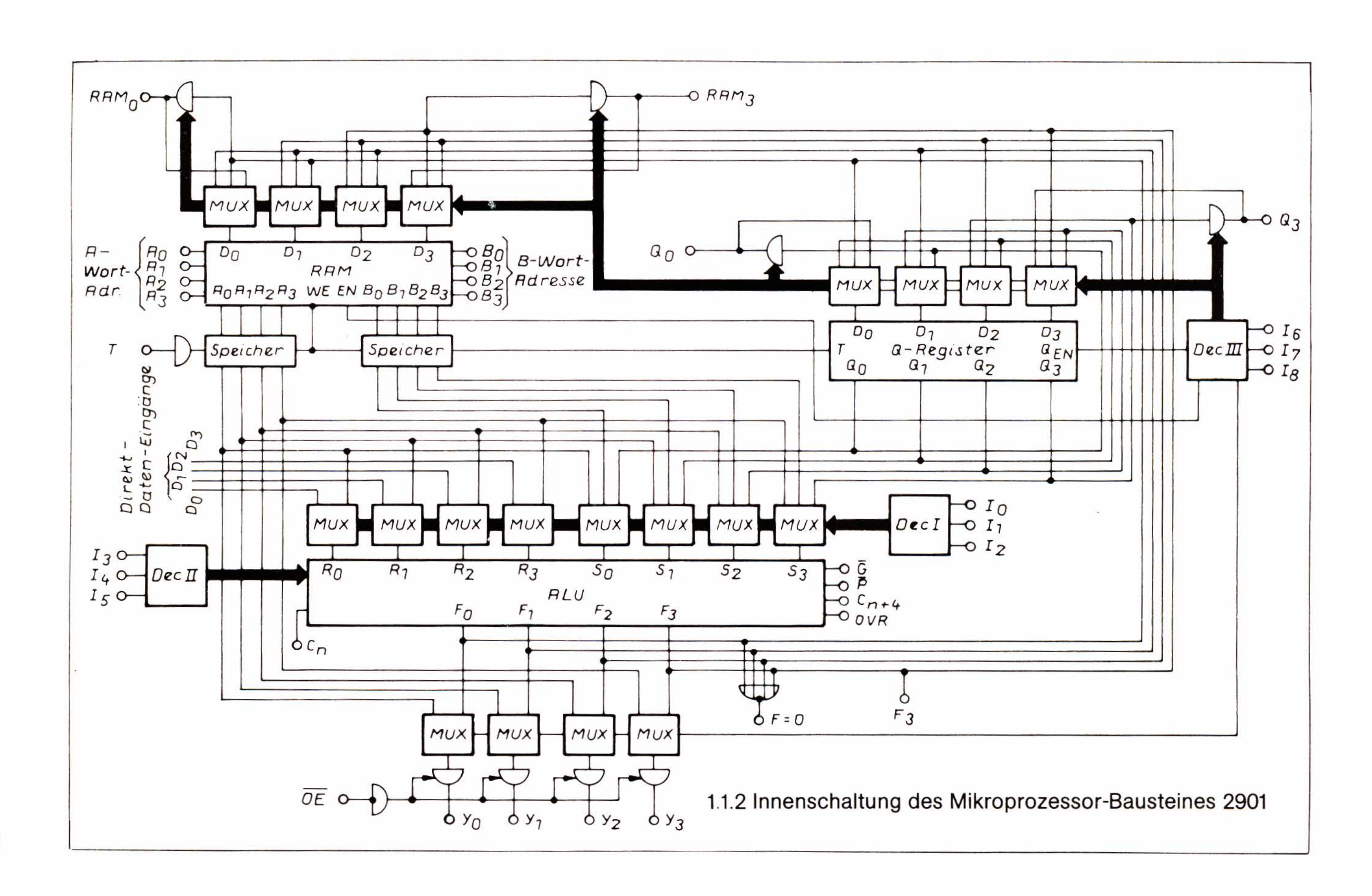
1.1.1 Blockschaltbild des Mikroprozessor-Bausteines 2901

Der Schreib-Lese-Speicher (RAM) des Bausteins 2901 kann die Daten zwischenspeichern. Je nach Adresse wird einer der Speicherplätze gewählt. Die Eingangsdaten für das Register B erhält das RAM von einem Schieberegister (RAM-Shift). In dem Schieberegister können die eingeschriebenen Daten um je eine Stelle nach rechts oder links geschoben werden. Die so verschobenen Daten speichern wir in dem RAM auf einer Adresse zwischen. Der Mikroprozessor kann diese Daten nach Bedarf abrufen. Dabei passieren die Daten den Daten-Selektor, die ALU und stehen dann wieder an dem RAM-Shift an. Hier können die Daten wieder um eine Stelle verschoben und auf einer Adresse abgespeichert werden. Aus dieser Verschiebung kann eine Multiplikation oder eine Division in Verbindung mit der ALU entstehen.
Neben dem RAM hat der Baustein 2901 ein Q-Register. Das Q-Register ist ein einfaches Zwischenregister. Hier wird eine Information kurz zwischengespeichert und bei Bedarf abgerufen. Die F-Eingänge sind mit den F-Ausgängen der ALU verbunden. Die Ausgangsdaten der ALU können zwischengespeichert werden. Dies ist z. B. bei Kettenrechnungen erforderlich. Vor dem Q-Register befindet sich ein Schieberegister (Q-Shift). Mit diesem Schieberegister lassen sich die Ausgangsdaten des Q-Registers um eine Stelle nach rechts oder links schieben. Damit ergibt sich innerhalb des Q-Registers eine Multiplikations- oder Divisionsmöglichkeit.

Die beiden Schieberegister haben kombinierte serielle Ein- und Ausgänge. Über diese können Daten seriell eingeschrieben oder ausgelesen werden.
In Bild 1.1.2 ist die Innenschaltung des Bausteines 2901 gezeigt. Die ALU erhält die beiden Eingangsdatenwörter von den acht vorgeschalteten Multiplexern (MUX). Die Multiplexer werden durch den Decoder Dec I angesteuert. Es ergibt sich folgende Funktionstabelle:

Mikrocodewort				Eingänge	
Nr.	I_2	I_1	I_0	R	S
0	L	L	L	A	Q
1	L	L	H	A	B
2	L	H	L	Ø	Q
3	L	H	H	Ø	B
4	H	L	L	Ø	A
5	H	L	H	D	A
6	H	H	L	D	Q
7	H	H	H	D	Ø

A = Daten aus dem RAM A
B = Daten auf dem RAM B
Q = Daten aus einem Zwischenregister Q
D = Daten von dem direkten Eingang
Ø = Daten aus der 0-Logik



1.1.2 Innenschaltung des Mikroprozessor-Bausteines 2901

Die Daten des Bausteines 2901 sind über den Ausgangsdaten-Selektor erhältlich. Die Ausgangsdaten werden durch den A- oder F-Eingang gebildet. Mit dem A-Eingang ist der Ausgang des A-RAMs mit dem Y-Ausgang verbunden. Bei dem F-Ausgang sind die Ausgangsfunktionen an dem Y-Ausgang.

Mit den vorgeschalteten Multiplexern wird die Informationssteuerung für die ALU-Eingänge R und S bestimmt. Je nach Mikrocodewort ergibt sich eine andere Verschaltung.
Die einzelnen Daten werden in der ALU verarbeitet. Maßgebend für die ALU sind die Befehle an den Decoder Dec II. Es ergibt sich folgende Funktionstabelle:

Mikrocodewort				ALU	
Nr.	I_5	I_4	I_3	Funktion	Symbol
0	L	L	L	R plus S	$R + S$
1	L	L	H	S minus R	$S - R$
2	L	H	L	R minus S	$R - S$
3	L	H	H	R oder S	$R \vee S$
4	H	L	L	R und S	$R \wedge S$
5	H	L	H	$\overline{R}$ und S	$\overline{R} \wedge S$
6	H	H	L	R ex - or S	$R \forall S$
7	H	H	H	R in - or S	$\overline{R \forall S}$

Mit den drei arithmetischen und den fünf logischen Funktionen lassen sich zahlreiche Rechen- und Logik-Operationen durchführen.

Mit dem Decoder Dec III können weitere Funktionen mit dem RAM und Q-Register durchgeführt werden, wie die nachfolgende Funktionstabelle zeigt:

Mikrocodewort				RAM-Funktion		Q-Register-Funktion	
Nr.	I_8	I_7	I_6	Schieben	Laden	Schieben	Laden
0	L	L	L	X	keine	keine	$F \rightarrow Q$
1	L	L	H	X	keine	X	keine
2	L	H	L	keine	$F \rightarrow B$	X	keine
3	L	H	H	keine	$F \rightarrow B$	X	keine
4	H	L	L	abwärts	$F/2 \rightarrow B$	abwärts	$Q/2 \rightarrow B$
5	H	L	H	abwärts	$F/2 \rightarrow B$	X	keine
6	H	H	L	aufwärts	$2F \rightarrow B$	aufwärts	$2Q \rightarrow Q$
7	H	H	H	aufwärts	$2F \rightarrow B$	X	keine

X = Ausgang hochohmig (Tri-State-Verhalten)

Der Decoder Dec III steuert auch die Ausgänge Y, die seriellen RAM-Ein- und -Ausgänge und die seriellen Q-Ein- und -Ausgänge. Es ergibt sich folgende Funktionstabelle:

Mikro-codewort				Y	RAM		Q	
Nr.	I_8	I_7	I_6	Ausgang	RAM_0	RAM_3	Q_0	Q_3
0	L	L	L	F	X	X	X	X
1	L	L	H	F	X	X	X	X
2	L	H	L	A	X	X	X	X
3	L	H	H	F	X	X	X	X
4	H	L	L	F	F_0	IN_3	Q_0	IN_3
5	H	L	H	F	F_0	IN_3	Q_0	X
6	H	H	L	F	IN_0	F_3	IN_0	Q_3
7	H	H	H	F	IN_0	F_3	X	Q_3

X = Ausgang hochohmig (Tri-State-Verhalten)

Die beiden Punkte RAM_0 und RAM_3 in Bild 1.1.2 sind Ein- und Ausgänge. Hat die Bezeichnung in der Funktionstabelle ein F_0, so sind die Ausgangsdaten der ALU direkt auf den Ausgang RAM_0 geschaltet. Gleichzeitig werden die Daten der ALU auf den adressierten Speicherplatz eingeschrieben. Der Ausgang F_0 der ALU liegt seriell an dem RAM_0-Ausgang. Mit der Bezeichnung IN_0 wird die Eingangsinformation des RAM_0-Einganges auf den adressierten Speicherplatz eingeschrieben.

In Verbindung mit den beiden Befehlseingängen des Decoders Dec I und II ergeben sich folgende logische Befehle:

	Oktal-Code		
Funktion	I_{543}	I_{210}	Symbol
UND	4	0	$A \wedge Q$
	4	1	$A \wedge B$
	4	5	$D \wedge A$
	4	6	$D \wedge Q$
ODER	3	0	$A \vee Q$
	3	1	$A \vee B$
	3	5	$D \vee A$
	3	6	$D \vee Q$
Exklusiv-ODER	6	0	$A \veebar Q$
	6	1	$A \veebar B$
	6	5	$D \veebar A$
	6	6	$D \veebar Q$
Inklusiv-ODER	7	0	$\overline{A \veebar Q}$
	7	1	$\overline{A \veebar B}$
	7	5	$\overline{D \veebar A}$
	7	6	$\overline{D \veebar Q}$
NICHT	7	2	$\overline{Q}$
	7	3	$\overline{B}$
	7	4	$\overline{A}$
	7	7	$\overline{D}$

	Oktal-Code		
Funktion	I_{543}	I_{210}	Symbol
keine	6 6 6 6	2 3 4 7	Q B A D
keine	3 3 3 3	2 3 4 7	Q B A D
NULL-setzen	4 4 4 4	2 3 4 7	0 0 0 0
Implikation	5 5 5 5	0 1 5 6	$\overline{A} \wedge Q$ $\overline{A} \wedge B$ $\overline{D} \wedge A$ $\overline{D} \wedge Q$

Der Oktal-Code ist in die entsprechenden Signale umzuwandeln. Die logischen Befehle sind immer pro Bit hergestellt, d. h. es werden zwei gleichwertige Eingänge verknüpft. Der Übertragseingang C_n bleibt wirkungslos.

Für die arithmetischen Rechenoperationen ergeben sich zwei Funktionstabellen, da die einzelnen Tabellen von dem Signal des Übertragseinganges C_n abhängen:

$C_n = 0$ (L-Signal):

	Oktal-Code		
Funktion	I_{543}	I_{210}	Symbol
Addiere	0 0 0 0	0 1 5 6	$A+Q$ $A+B$ $D+A$ $D+Q$
keine	0 0 0 0	2 3 4 7	Q B A D
Verringere	1 1 1 2	2 3 4 7	$Q-1$ $B-1$ $A-1$ $D-1$
Komplement	2 2 2 1	2 3 4 7	$-Q-1$ $-B-1$ $-A-1$ $-D-1$

	Oktal-Code		
Funktion	I_{543}	I_{210}	Symbol
2^{n-1} Subtraktion	1	0	Q − A − 1
	1	1	B − A − 1
	1	5	A − D − 1
	1	6	Q − D − 1
	2	0	A − Q − 1
	2	1	A − B − 1
	2	5	D − A − 1
	2	6	D − Q − 1

$C_n = 1$ (H-Signal) :

	Oktal-Code		
Funktion	I_{543}	I_{210}	Symbol
Addiere und + 1	0	0	A + Q + 1
	0	1	A + B + 1
	0	5	D + A + 1
	0	6	D + Q + 1
Erhöhe	0	2	Q + 1
	0	3	B + 1
	0	4	A + 1
	0	7	D + 1

	Oktal-Code		
Funktion	I_{543}	I_{210}	Symbol
keine	1	2	Q
	1	3	B
	1	4	A
	2	7	D
2^n Komplement	2	2	− Q
	2	3	− B
	2	4	− A
	1	7	− D
2^n Subtraktion	1	0	Q − A
	1	1	B − A
	1	5	A − D
	1	6	Q − D
	2	0	A − Q
	2	1	A − B
	2	5	D − A
	2	6	D − Q

Die Addition in dem Mikroprozessor 2901 hängt von dem Befehl ab, welches Register mit welchem addiert werden soll. Hat der Übertrag eine 0, also ein L-Signal, erhalten wir eine direkte Addition. Hat der Übertrag eine 1, also ein H-Signal, wird die Addition um eine 1 noch zusätzlich erhöht.

Bei einer Verringerung (decrement) eines Inhaltes wird von diesem eine 1 subtrahiert. Bei einer Erhöhung (increment) wird der Inhalt um eine 1 erhöht. Dies geschieht mittels einer Addition durch den Übertrag.
Das 2^n- und 2^{n-1}-Komplement ist für spezielle Funktionen geeignet. Das 2^n-Komplement ist folgendermaßen aufgebaut:

$$
\begin{aligned}
0 + 10 &= 10\\
1 + 9 &= 10\\
2 + 8 &= 10\\
&\dots\\
9 + 1 &= 10\\
10 + 0 &= 10
\end{aligned}
$$

Das 2^{n-1}-Komplement ist folgendermaßen aufgebaut:

$$
\begin{aligned}
0 + 9 &= 9\\
1 + 8 &= 9\\
2 + 8 &= 9\\
&\dots\\
8 + 1 &= 9\\
9 + 0 &= 9
\end{aligned}
$$

Bei den beiden Komplementen wird immer die Zahl hergestellt, die auf 10 bzw. 9 fehlt.
Die Subtraktion wird entweder im 2^n- oder im 2^{n-1}-Komplement durchgeführt. Für das 2^n-Komplement ist ein Übertrag erforderlich. Es ergeben sich folgende Rechenoperationen für den Dual-Code:

2^n-Verfahren

$$
\begin{array}{rlr}
5 \triangleq & 0101 \geqq & 0101\\
-2 \triangleq & 0010 \geqq + & 1101\\
\hline
 & & 1 \leftarrow \text{Übertrag}\\
\hline
3 & & 1\ \underbrace{0011}_{3}
\end{array}
$$

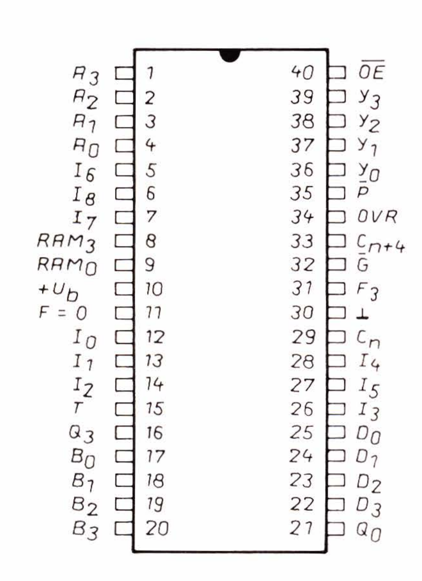
Beim 2^n-Verfahren wird die 2 komplett invertiert. Danach erfolgt die Addition, zusammen mit dem Übertrag. Das Ergebnis (Differenz) erscheint sofort an dem Ausgang F der ALU.

2^{n-1}-Verfahren

$$
\begin{array}{rlr}
5 \triangleq & 0101 \geqq & 0101\\
-2 \triangleq & 0010 \geqq + & 1110\\
\hline
3 & & 1\ \underbrace{0011}_{3}
\end{array}
$$

Beim 2^{n-1}-Verfahren wird von der 2 zuerst eine 1 subtrahiert. Danach erfolgt die Negation. Dadurch kann der Übertrag eine 0 aufweisen. Hat in diesem Fall der Übertrag eine 1, wird die Differenz um eine 1 erhöht (incrementiert).
Beim Programmieren des Mikrocomputers können wir selbst entscheiden, welche Subtraktion wir vorziehen. Im Bild 1.1.3 ist das Anschlußbild für den Mikroprozessor 2901 gezeigt. Für die einzelnen Pinanschlüsse ergeben sich folgende Definitionen:

A_{0-3}: Mit den vier Adresseneingängen können wir den internen Schreib-Lese-Speicher, das RAM A, adressieren. Die Ausgänge F der



1.1.3 Anschlußschema des Mikroprozessor-Bausteines 2901

ALU sind über den Eingangsmultiplexer mit dem adressierten Speicherplatz verbunden. Insgesamt stehen 16 Speicherplätze zur Verfügung.

B_{0-3}: Mit den vier Adresseneingängen können wir den internen Schreib-Lese-Speicher, das RAM B, adressieren. Die Ausgänge F der ALU sind ebenfalls über den Eingangsmultiplexer mit dem adressierten Speicherplatz verbunden. Insgesamt stehen 16 Speicherplätze zur Verfügung.

I_{0-8}: Über diese neun Eingänge werden die Befehle auf den Mikroprozessor gegeben. Wir unterscheiden zwischen den Eingängen I_{012}, I_{345} und I_{678}. Die Eingänge I_{012} steuern den Decoder Dec I an. Mit einem Befehl bestimmen wir, welche Register für die ALU zusammengeschaltet werden. Die Eingänge I_{345} steuern den Decoder Dec II an. Mit einem Befehl bestimmen wir die Rechenoperation in der ALU. Die Eingänge I_{678} steuern den Decoder Dec III an. Mit einem Befehl bestimmen wir die RAM-, Q-Register- und Y-Ausgangsfunktion.

Q_3, RAM_3: Mit diesen beiden Ein- und Ausgängen werden die MSB (most significant bit or byte ≙ höherwertigen Bit oder Byte) verarbeitet. In unserem Fall werden die höherwertigen

Bits ein- oder ausgegeben. Bei der Datenausgabe schaltet der Tri-State-Ausgang des Mikroprozessors das Ausgangssignal der ALU auf den Anschlußpunkt. Eine Speicherung auf dem adressierten Speicherplatz in dem RAM oder eine Zwischenspeicherung in dem Q-Register erfolgt gleichzeitig mit den anderen Ausgangsbits der ALU. Beim Einschreiben eines MSB über einen der beiden Anschlußpunkte hat zur Folge, daß der Tri-State-Ausgang des Mikroprozessors hochohmig ist. Der betreffende Eingang wird mit der adressierten RAM-Speicherzelle oder mit dem Q-Register verbunden.

Q_0, RAM_0: Mit diesen beiden Ein- und Ausgängen werden die LSB (Least significant bit or byte ≙ niederwertiges Bit oder Byte) verarbeitet. In unserem Fall werden die niederwertigen Bits eingeschrieben oder ausgelesen. Bei der Datenausgabe schaltet der Tri-State-Ausgang des Mikroprozessors das Ausgangssignal der ALU auf den Anschlußpunkt. Eine Speicherung auf dem adressierten Speicherplatz in dem RAM oder eine Zwischenspeicherung in dem Q-Register erfolgt gleichzeitig mit den anderen Ausgangsbits der ALU. Beim Einschreiben eines LSB über einen der beiden Anschlußpunkte hat zur Folge, daß der Tri-State-Ausgang des Mikroprozessors hochohmig ist. Der betreffende Eingang wird mit der adressierten RAM-Speicherzelle oder mit dem Q-Register verbunden.

D_{0-3}: Über diese vier Eingänge können die direkten Daten in die ALU eingegeben werden. Dabei ist D_0 das LSB und D_3 das MSB.

Y_{0-3}: An diesen Ausgangspunkten erhalten wir die Daten der ALU oder des RAMs A. Erfolgt keine Ansteuerung der Tri-State-Ausgänge durch den Decoder Dec III, so sind die Ausgänge hochohmig. Bei einer Ansteuerung, schalten sie die Informationen durch.

$\overline{OE}$: Der $\overline{OE}$-Eingang ist der „output enable“ oder die Ausgangssperre. Hat dieser Eingang ein H-Signal, so sind die Y-Ausgänge des Mikroprozessors hochohmig. Bei einem L-Signal schalten die Ausgänge die Informationen direkt durch.

$\overline{P}$, $\overline{G}$: Die beiden Ausgänge dienen zur Erweiterung des Mikroprozessor-Systems. Bei diesem Mikrocomputer-Modell sind die Ausgänge separat herausgeführt, haben aber keine Funktion.

OVR: Mit diesem OVERFLOW oder Überlauf erkennt der Mikroprozessor, daß bei einer Rechenoperation an der fünften Stelle ein

Bit aufgetreten ist. Der Mikroprozessor 2901 ist ein erweiterbarer 4-Bit-Baustein. Über den OVR-Ausgang kann in Verbindung mit den $\overline{P}$- und $\overline{G}$-Eingang das System auf 8-, 12-, 16- Bits usw. erweitert werden.

F = 0: Wenn wir uns das Bild 1.1.2 betrachten, so sehen wir ein NOR-Gatter, das diese Ausgangsfunktion erzeugt. Der Ausgang hat erst dann ein H-Signal, wenn an der ALU ein 0-Wort, d. h. lauter L-Signale erzeugt worden sind. Hat nur einer dieser ALU-Ausgänge ein H-Signal, geht der = 0-Ausgang auf L-Signal.

C_n: An diesem Punkt erfolgt die Eingabe des Übertrags (CARRY) in den Mikroprozessor 2901.

C_{n+4}: Tritt bei einer Rechenoperation ein Übertrag auf, so wird dieser an dem Ausgang angezeigt.

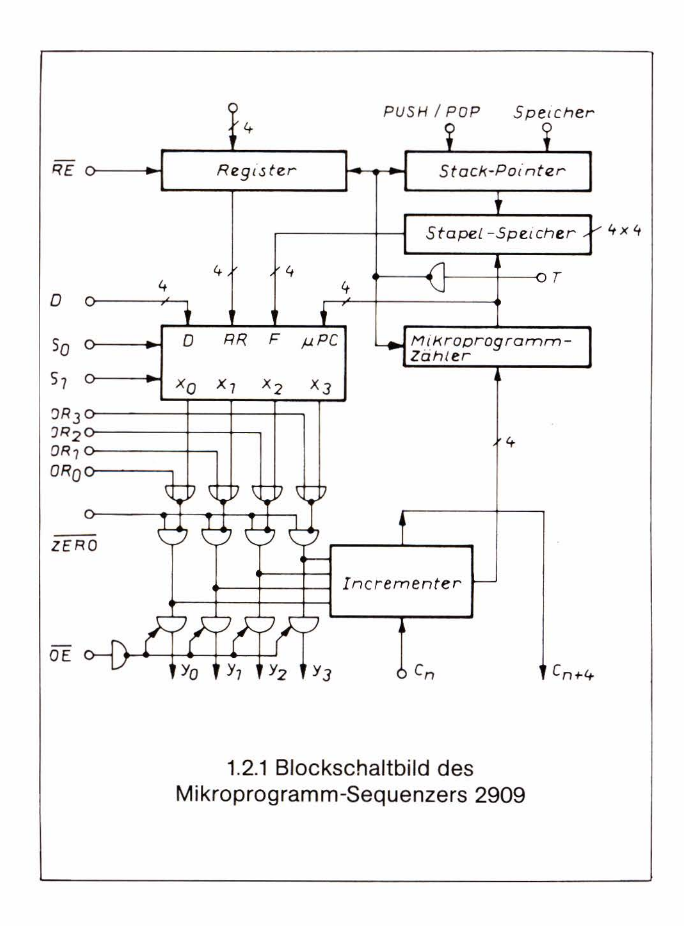
F_3: Mit diesem Ausgang wird das MSB überwacht. Hat die ALU an dem Ausgang F_3 ein Signal, so wird dies direkt zu dem Ausgang übertragen.

T: Takteingang für den Mikroprozessor 2901. Das Q-Register und die Speicher nach dem RAM reagieren auf einen positiven Taktimpuls. Hat der Takt ein L-Signal, so werden die adressierten Speicherplätze nicht in den Speicher eingeschrieben. Erst bei einer positiven Taktflanke erfolgt die Datenübernahme. Danach können sich die Adressen des RAMs wieder ändern, ohne daß sich der Speicherinhalt ändert.

1.2 Mikroprogramm-Sequenzer 2909

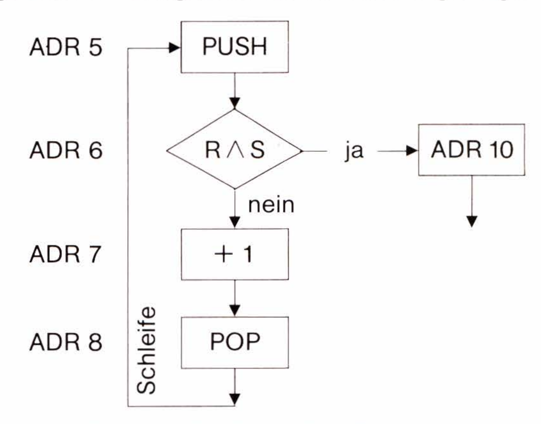
Für den Betrieb des Mikroprozessors ist der Mikroprogramm-Sequenzer erforderlich. In Bild 1.2.1 ist das Blockschaltbild für den Baustein 2909 gezeigt. Der Stack-Pointer ist der Kellerspeicher des Mikrocomputer-Systems. Wenn wir einen Sprung in ein Unterprogramm programmieren, wird in dem Stack-Pointer die letzte Adresse festgehalten. Mit dem PUSH/POP-Eingang können wir durch einen Befehl die Adresse in dem internen Speicher hinterlegen (einschreiben) oder abrufen (auslesen). Der Speicher in dem Mikroprogramm-Sequenzer ist ein „Stapel-Speicher“. Gibt der Programmierer den Befehl „PUSH“ in sein Programm, so wird bei dem PUSH die letzte Programmadresse in den Stapel-Speicher abgelegt. Danach arbeitet der Mikroprozessor weiter. Erscheint nun in dem Programm ein POP-Befehl, springt der Mikroprozessor auf PUSH zurück, holt sich die letzte Programmadresse und beginnt wieder von neuem.

PUSH und POP verwendet der Programmierer bei einem Schleifenprogramm. Innerhalb der Schleife be-



1.2.1 Blockschaltbild des Mikroprogramm-Sequenzers 2909

findet sich eine Logikfunktion mit einer Sprungadresse. Der Mikroprozessor arbeitet solange in der Schleife, bis die Logikfunktion erfüllt ist. An der Logikfunktion hängt eine Adresse. Mit dieser Adresse kann der Mikroprozessor das Schleifenprogramm verlassen.
Beim PUSH-Befehl wird eine Programmschleife eingeleitet, d. h. die letzte Adresse im Stapelspeicher abgelegt. Mit dem POP holt sich der Mikroprozessor diese Adresse und beginnt sein Programm wieder. In dem nachfolgenden Flußdiagramm ist der Ablauf gezeigt:



Mit PUSH wird die Schleife eingeleitet. Danach erfolgt der logische Vergleich durch ein UND-Gatter. Ist die UND-Bedingung nicht erfüllt, springt der Mikroprozessor auf die Adresse ADR 7 und erhöht seinen Speicher-

inhalt um eine Eins. Mit dem POP holt er sich die gestapelte Adresse 6 aus dem Stack-Pointer und erhält so seinen nächsten Befehl. Er vergleicht die beiden Register und trifft eine Entscheidung. Bei einem negativen Bescheid erhöht der Programmzähler auf ADR 7 und es erfolgt eine weitere Addition mit einer Eins. Ist der Bescheid positiv, erhält der Mikroprozessor die ADR10 und es wird ein bedingter Sprung eingeleitet.

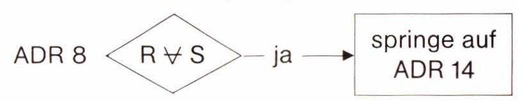
Der Stapel-Speicher wird auch von dem Mikroprogramm-Zähler angesteuert. In dem Mikroprogramm-Zähler erfolgt die Adressierung des Mikrocomputer-Systems. Der Zähler beginnt bei der Adresse ADR 0, Diese Adresse liegt über den µPC-Eingang an dem Multiplexer. Der Multiplexer schaltet die Adresse über die NOR-Gatter und UND-Gatter auf die Tri-State-Ausgänge. Gleichzeitigt liegt die Adresse 0 über dem Incrementer (Erhöher) an dem Mikroprogramm-Zähler. Dabei wird die Adresse 0 in dem Incrementer um eine Eins erhöht und an dem Zähler liegt die Adresse 1. Bei einem Signal des Taktes T übernimmt der Zähler die Adresse ADR 1 und speichert sie. Liegt die Adresse an den Tri-State-Ausgängen, wird durch ein Signal der Multiplexer gesperrt. Wurde der Befehl auf der Adresse ADR 0 von dem Mikroprozessor durchgeführt, meldet er dies dem Mikroprogramm-Sequenzer. Damit wird der Inhalt des Zählers, also die ADR 1 auf die Ausgänge geschaltet. Gleichzeitig erhöht der Incrementer die Adresse auf ADR 2 und speichert diese bei dem nächsten Taktimpuls im Mikroprogramm-Zähler. Diesen Arbeitszyklus bezeichnet man als den „Neumann-Zyklus“, d. h.:

a) der Mikroprozessor, bei dem Mikrocomputer-System der Mikroprogramm-Sequenzer, sendet eine Adresse aus.
b) mit dieser Adresse wird ein Befehl geholt.
c) der Befehl liegt an dem Mikroprozessor 2901 an, wird decodiert, also entschlüsselt.
d) der Mikroprozessor führt den decodierten Befehl aus.
e) er meldet dem Mikroprogramm-Sequenzer, daß der Befehl ausgeführt wurde und dieser gibt eine neue Adresse frei.

Nach diesem Schema arbeitet jedes Mikroprozessor- und Mikrocomputer-System.

Der Mikroprogramm-Sequenzer ist der Denker in diesem Mikrocomputer-System. Er gibt die einzelnen Adressen an den Mikroprozessor weiter und damit kann dieser arbeiten.

In dem Multiplexer von Bild 1.2.1 wird durch die vier Eingänge der Programmablauf eines Prozesses sichergestellt. Der D-Eingang liefert die direkten Adressen. Diese Adressen sind an einen Befehl angehängt. Wir sprechen hierbei von einer Sprung-Adresse:



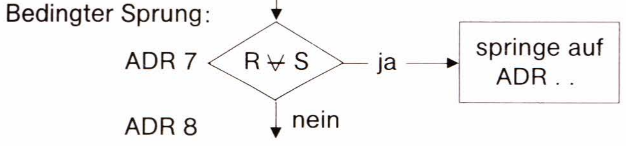
Auf der ADR 8 in einer Programmierung befindet sich ein Exklusiv-ODER-Befehl mit einer angehängten Adresse. Ist die Verknüpfung positiv, so gibt der Mikroprozessor die Adresse ADR 14 auf die direkten Adressen-Eingänge des Multiplexers. Damit erhält der Mikroprozessor seine Sprungadresse, d. h. er springt von der ADR 8 auf die ADR 14. Die ADR 14 wird durch den Multiplexer auf die Y-Ausgänge von Bild 1.2.1 geschaltet. Man bezeichnet diesen Sprung als bedingten Sprung, da er von einer logischen Entscheidung abhängt.

Über den AR-Eingang erhält der Mikroprogramm-Sequenzer seine unbedingte Sprung-Adresse. Diese Adresse ist von keiner logischen Entscheidung abhängig, sondern diese legt der Programmierer im Programm fest.

ADR 6 | springe auf ADR 12 |

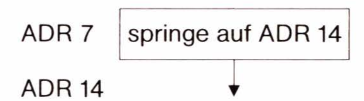
Über das Register erhält der Multiplexer diese Adresse ADR 12. Der Mikrocomputer springt von ADR 6 unbedingt auf ADR 12. Das „springe auf" ist ein unbedingter Sprung-Befehl. Mit der angehängten Adresse erhält der Mikroprozessor seine nächste Adresse.

Bedingter Sprung:



Auf der Adresse 7 wird eine Entscheidung getroffen. Ist die Entscheidung negativ, setzt der Mikrocomputer sein Programm auf der ADR 8 weiter. Ist die Entscheidung positiv, springt der Mikrocomputer auf die ADR und setzt dort sein Programm weiter.

Unbedingter Sprung:



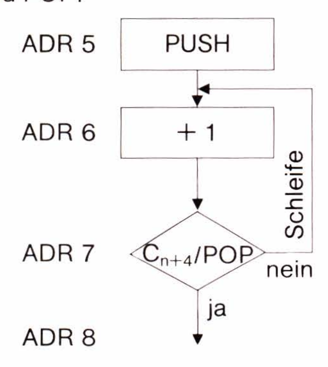
Auf der Adresse ADR 7 gibt der Befehl einen unbedingten Sprung frei. Der Mikroprozessor überspringt in seinem Programm die Adressen ADR 8 bis 13 und setzt sein Programm auf ADR 14 fort.

Die einzelnen Adressen stehen an dem Incrementer an. Der Incrementer erhöht diese Adresse immer um eine Eins. Damit kann der Mikrocomputer nach der Befehlsausführung sein Programm auf seiner erhöhten Adresse fortführen.

Über den F-Eingang des Multiplexers von Bild 1.2.1 erhält der Mikroprozessor seine D-Adresse aus dem Stapel-Speicher. In dem Stapel-Speicher befindet sich die letzte Adresse vor einem Unterprogrammsprung. Das Mikrocomputer-Modell bietet uns zwei Möglichkeiten eines Unterprogrammsprunges. Wir können mit einem PUSH die beiden Möglichkeiten einleiten. Durch den POP-Befehl haben wir die Möglichkeit einer Schleifenunterbrechung und Programmverlauf auf der näch-

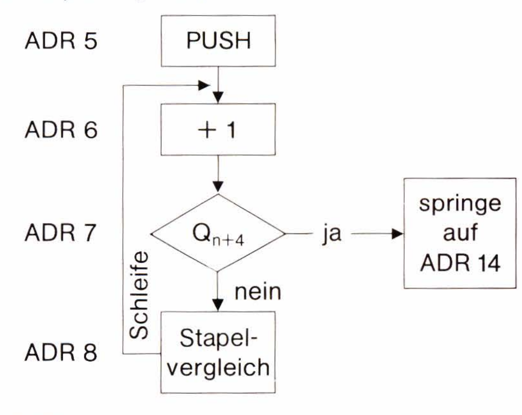
sten Adresse oder eine Schleifenunterbrechung durch einen Befehl in der Schleife. Hierzu ist aber am Ende der Schleife ein Stapelvergleich notwendig.

PUSH und POP:



Mit dem PUSH begeben wir uns mit dem Programm in eine Schleife. Die ADR 5 wird erhöht, also mit ADR 6 im Stapel-Speicher gespeichert. Gleichzeitig wird zu dem Zustand in der ALU des Mikroprozessors 2901 eine Eins hinzuaddiert. Auf der Adresse ADR 7 wird eine Entscheidung getroffen. Ist die Bedingung C_{n+4} durch den Mikroprozessor erfüllt, wird das Programm auf der ADR 8 fortgesetzt. Der Mikroprozessor verläßt durch diese Entscheidung seine Schleife automatisch. Ist die Bedingung C_{n+4} aber nicht erfüllt, erscheint in dem Befehl der POP. Der Stapel-Speicher schaltet die letzte Adresse auf den Mikroprozessor und damit auf ADR 6. In diesem Befehl wird eine Addition mit einer Eins in der ALU durchgeführt. Danach kommt in dem Programm wieder die Entscheidung. Ist C_{n+4} nicht erfüllt, springt der Mikrocomputer auf die ADR 6 zurück und erhöht. Der Vorgang von POP dauert so lange, bis C_{n+4} erfüllt ist. Durch POP wird auf die gespeicherte Adresse nach PUSH zurückgesprungen.

PUSH und Stapelvergleich:



Mit dem PUSH begeben wir uns mit dem Programm in eine Schleife. Die ADR 5 wird erhöht in dem Stapel-Speicher festgehalten. Auf der Adresse ADR 6 erfolgt eine Addition mit einer Eins. Auf der ADR 7 wird eine Entscheidung getroffen. Ist die Entscheidung negativ ausgefallen, so wird auf der Adresse ADR 8 ein Stapel-

vergleich vorgenommen. Der Mikroprozessor springt dadurch unbedingt auf ADR 6 zurück und erhöht den Wert des ALU-Inhaltes um eine Eins. Ist die Entscheidung positiv, springt der Mikrocomputer auf die ADR 14 und führt dort sein Programm weiter.
Der Unterschied zwischen POP und dem Stapelvergleich ist die Möglichkeit eines bedingten Programmsprunges in der Programmschleife des Stapelvergleiches. Durch die ADR 7 kann er sofort auf die Adresse ADR 14 springen. Bei dem POP sparen wir uns dagegen eine Adresse, aber das Programm kann nur auf der nachfolgenden Adresse weitergeführt werden. Es ist kein unbedingter Sprung innerhalb der Schleife möglich.
In Bild 1.2.2 ist die Innenschaltung des Mikroprogramm-Sequenzers gezeigt. Über die beiden Steuereingänge S_0 und S_1 steuern wir den Multiplexer des Bausteines. Es ergibt sich folgende Funktionstabelle:

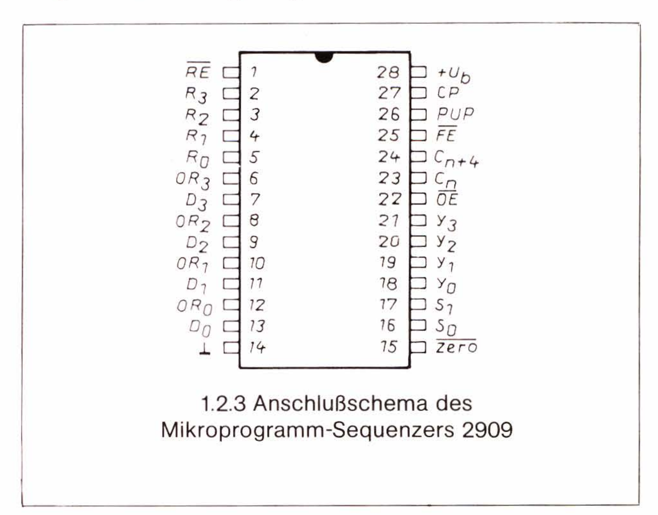
S_1	S_0	Funktion an Y
L	L	Mikroprogramm-Zähler µPC
L	H	Register REG
H	L	Stapel-Speicher SP
H	H	Direkte Adresse DA

Mit dem Eingang ZERO kann die Steuerung durch die beiden Eingänge S_0 und S_1 noch zusätzlich beeinflußt werden. Es ergibt sich folgende Funktionstabelle:

ZERO	S_1	S_0	Funktion an Y
L	L	L	keine
L	L	H	keine
L	H	L	keine
L	H	H	keine
H	L	L	Mikroprogramm-Zähler µPC in Verbindung mit den OR-Eingängen
H	L	H	Register REG in Verbindung mit den OR-Eingängen
H	H	L	Stapel-Speicher SP in Verbindung mit den OR-Eingängen
H	H	H	Direkte Adresse DA in Verbindung mit den OR-Eingängen

1.2.2 Innenschaltung des Mikroprogramm-Sequenzers 2909

Die OR-Eingänge liegen über UND-Gatter an und können das NOR-Gatter beeinflussen. In diesem Modell sind sie mit Masse verbunden.
Der Ausgang des NOR-Gatters ist mit invertierenden Tri-State-Stufen verbunden. Hat der $\overline{OE}$-Eingang ein L-Signal, werden die Informationen der NOR-Gatter auf die Ausgänge geschaltet. Bei einem H-Signal sind die Ausgänge hochohmig.
Die R-Eingänge sind über UND-Gatter mit dem internen Register verbunden. Die Steuerung des Registers wird durch den „Register-Enable"-Eingang vorgenommen. Die Flip-Flops können gesetzt werden, wenn an dem Steuereingang ein L-Signal anliegt und an den R-Eingängen ein H-Signal, d. h. der Q-Ausgang hat ein H-Signal. Bei einem L-Signal an den R-Eingängen kippen die Flip-Flops zurück und die Q-Ausgänge haben ein L-Signal.
Die Q-Ausgänge des Registers sind mit den entsprechenden UND-Gattern verbunden. Die UND-Gatter zusammen bilden den Multiplexer. Aus fünf Eingangssignalen wird ein Ausgangssignal gebildet. Die UND-Gatter sind über NOR-Gatter zusammengefaßt und liegen an dem Incrementer.
Der Incrementer erhöht den Stand der NOR-Ausgänge immer um eine Eins, aber der C-Eingang muß auf L-Signal liegen. Die Ausgänge des Incrementers sind mit dem Mikroprogramm-Zähler verbunden. Der Zähler kann den Stand übernehmen und speichert ihn zwischen. Die Ausgänge des Zählers sind mit einer Schreib-Lese-Logik und mit den UND-Gattern des Multiplexers verbunden. Mit der Schreib-Lese-Logik wird der Stapel-Speicher angesteuert. Mit einem L-Signal an dem FE-Eingang wird die Schreib-Lese-Logik und der Stack-Pointer blockiert. Bei einem H-Signal kann der Stack-Pointer vor- oder rückwärtszählen und die Schreib-Lese-Logik kann diese Adressen in den Stapel-Speicher einschreiben oder auslesen.
In Bild 1.2.3 ist das Anschlußbild des Mikroprogramm-Sequenzers 2909 gezeigt.



1.2.3 Anschlußschema des Mikroprogramm-Sequenzers 2909

1.3.1 Sender- und Empfänger-Baustein 2907

1.3 Sender- und Empfänger-Baustein 2907

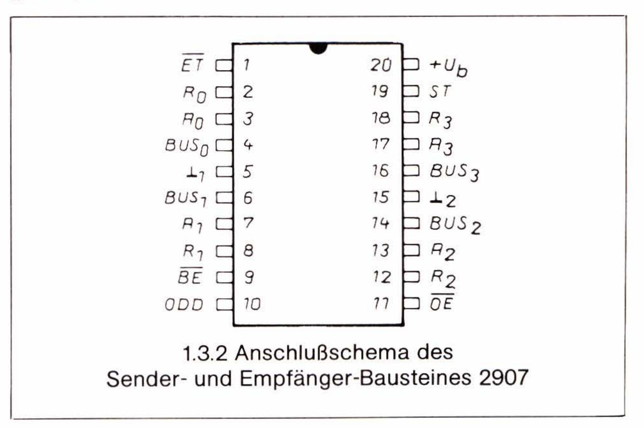
In Bild 1.3.1 ist die Innenschaltung des Sender- und Empfänger-Bausteines 2907 gezeigt. Über die vier A-Eingänge können die vier Sende-Flip-Flops gesetzt werden, wenn der Sende-Takt-Eingang ST von H- nach L-Signal kippt. Damit setzen oder kippen die Flip-Flops und an den Q-Ausgängen stehen die gespeicherten Signale. Über Tri-State-Ausgänge werden die gespeicherten Daten auf den BUS geschaltet. Dabei muß der $\overline{BE}$-Eingang auf L-Signal liegen. Der Baustein sendet. Über das BUS-System können auch Daten empfangen werden. Die BUS-Leitungen liegen über Buffer an den D-Eingängen der Empfänger-Flip-Flops. Die Informationen der BUS-Leitungen werden übernommen, wenn der $\overline{ET}$-Eingang auf L-Signal liegt. Die empfangenen Informationen werden in den Empfänger-Flip-Flops gespeichert.

Die gespeicherten Informationen in den Empfänger-Flip-Flops sind über invertierende Tri-State-Stufen mit den R-Ausgängen verbunden. Die gespeicherten Informationen können auf die Ausgänge geschaltet werden, wenn der $\overline{OE}$-Eingang ein L-Signal hat.

Die Eingangsinformationen an den A-Eingängen und die Ausgangsinformationen an den Q-Ausgängen der Empfänger-Flip-Flops können in der Parity-Logik überprüft werden. Die Parity-Logik bildet die Quersumme aus den Informationen. Ist die Quersumme gerade, so erscheint an dem Ausgang der Schaltung ein L-Signal. Bei einer ungeraden Quersumme hat der Ausgang ein H-Signal. Durch den $\overline{BE}$-Eingang wird die Parity-Logik angesteuert. Hat der $\overline{BE}$-Eingang ein H-Signal, so ist die Sendeeinrichtung gesperrt, aber die Parity-Logik für den Empfänger arbeitet. Mit einem L-Signal arbeitet die Sendeeinrichtung und die Parity-Logik erzeugt ein Signal für die Sendeinformationen.

Der Baustein 2907 kann mit der Parity-Logik die Quersumme aus den Sende- oder Empfangs-Daten bilden. Welche Parity-Logik verwendet wird, hängt von dem Signal an dem $\overline{BE}$-Eingang ab.

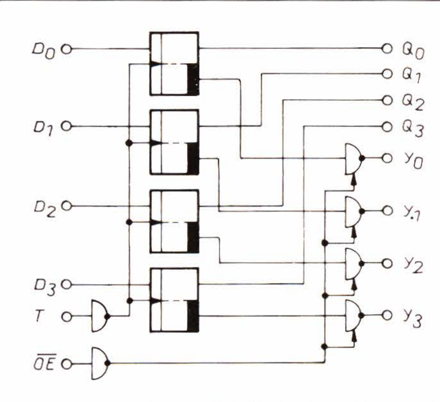
In Bild 1.3.2 ist das Anschlußbild des Bausteines 2907 gezeigt.



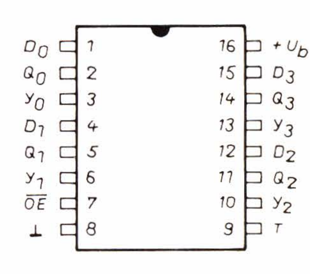
1.3.2 Anschlußschema des Sender- und Empfänger-Bausteines 2907

1.4 Register-Baustein mit Tri-State-Ausgängen 2918

In Bild 1.4.1 ist die Innenschaltung des Bausteines 2918 gezeigt. Der Baustein hat vier D-Flip-Flops. Die D-Flip-Flops werden an den D-Eingängen vorbereitet und mit einem positiven Taktimpuls angesteuert.
Die Q-Ausgänge der vier Flip-Flops bilden den Standard-Ausgang. An diesem Ausgang liegen immer die gespeicherten Informationen. Die $\overline{Q}$-Ausgänge sind mit den Tri-State-Buffern verbunden. In den Buffern werden die Eingangssignale negiert. Gesteuert werden die Buffer von dem $\overline{OE}$-Eingang. Hat dieser Eingang ein H-Signal, sind die Ausgänge der Buffer hochohmig. Mit einem L-Signal liegen die Eingangssignale invertiert an den Y-Ausgängen. In dem Baustein werden die Flags (Flaggen oder Zeichen) gespeichert. Mit den Flags kann der Mikroprozessor in seinem Programm bei einer Entscheidung einen bedingten Sprung einleiten. Die Flags sind die Ausgänge C_{n+4}, der F_3, der OVR und der F = 0.
Die Flags werden in dem Register-Baustein gespeichert. Die Standard-Ausgänge liefern uns gleichbleibende Daten. Die Tri-State-Ausgänge sind dagegen mit dem Daten-BUS verbunden. Die Flags können durch den BUS auf einen anderen Baustein geschaltet (transportiert) werden.
In Bild 1.4.2 ist das Anschlußbild für den Baustein 2918 gezeigt.



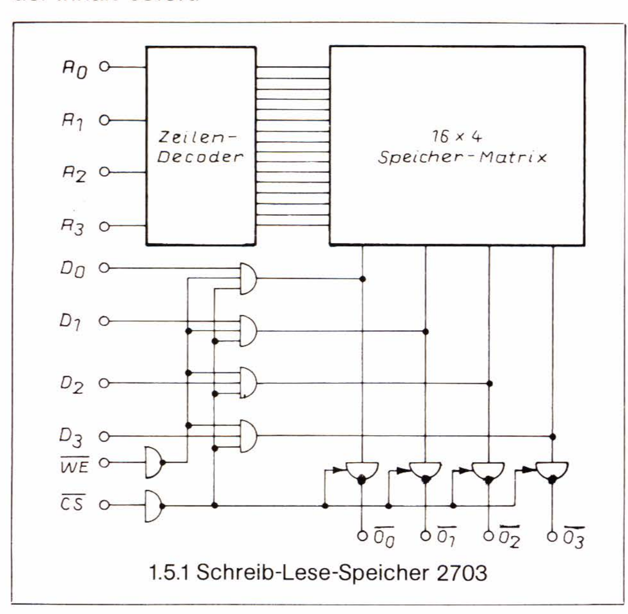
1.4.1 Register-Baustein mit Tri-State-Ausgängen 2918



1.4.2 Anschlußschema des Bausteines 2918

1.5 Schreib-Lese-Speicher 2703

Schreib-Lese-Speicher sind RAMs (random access memory), also Speicher mit einem wahlfreien Zugriff. In einem RAM werden Informationen eingeschrieben und bei Bedarf wieder zerstörungsfrei ausgelesen. Schalten wir die Betriebsspannung ab, verflüchtigt sich der Inhalt sofort.

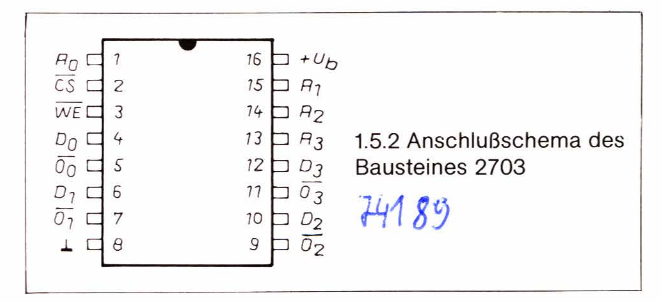


1.5.1 Schreib-Lese-Speicher 2703

In Bild 1.5.1 ist der Aufbau eines RAMs gezeigt. Über die A-Eingänge (Adressen) kann ein Speicherplatz adressiert werden. Da vier A-Eingänge vorhanden sind, hat das RAM 16 Speicherplätze. Jede einzelne Adresse wird in dem Zeilendecoder decodiert und liegt an einer der 16 Zeilen der Speicheranordnung.

Die Daten für die Speicheranordnung liegen an den D-Eingängen (Daten). Über die UND-Gatter kann der Schreib- oder Lese-Betrieb bestimmt werden. Die anstehenden Daten können eingeschrieben werden, wenn der $\overline{WE}$-Eingang und der $\overline{CS}$-Eingang je ein L-Signal haben. Die vier Eingänge liegen über das UND-Gatter an der adressierten Speicherzelle. Der Schreib-Vorgang ist beendet, wenn einer der beiden Steuereingänge ein H-Signal hat.

Der Lese-Vorgang läßt sich durchführen, wenn der $\overline{CS}$-Eingang ein L-Signal hat. Die eingeschriebenen Daten einer adressierten Speicherzelle liegen über die



1.5.2 Anschlußschema des Bausteines 2703

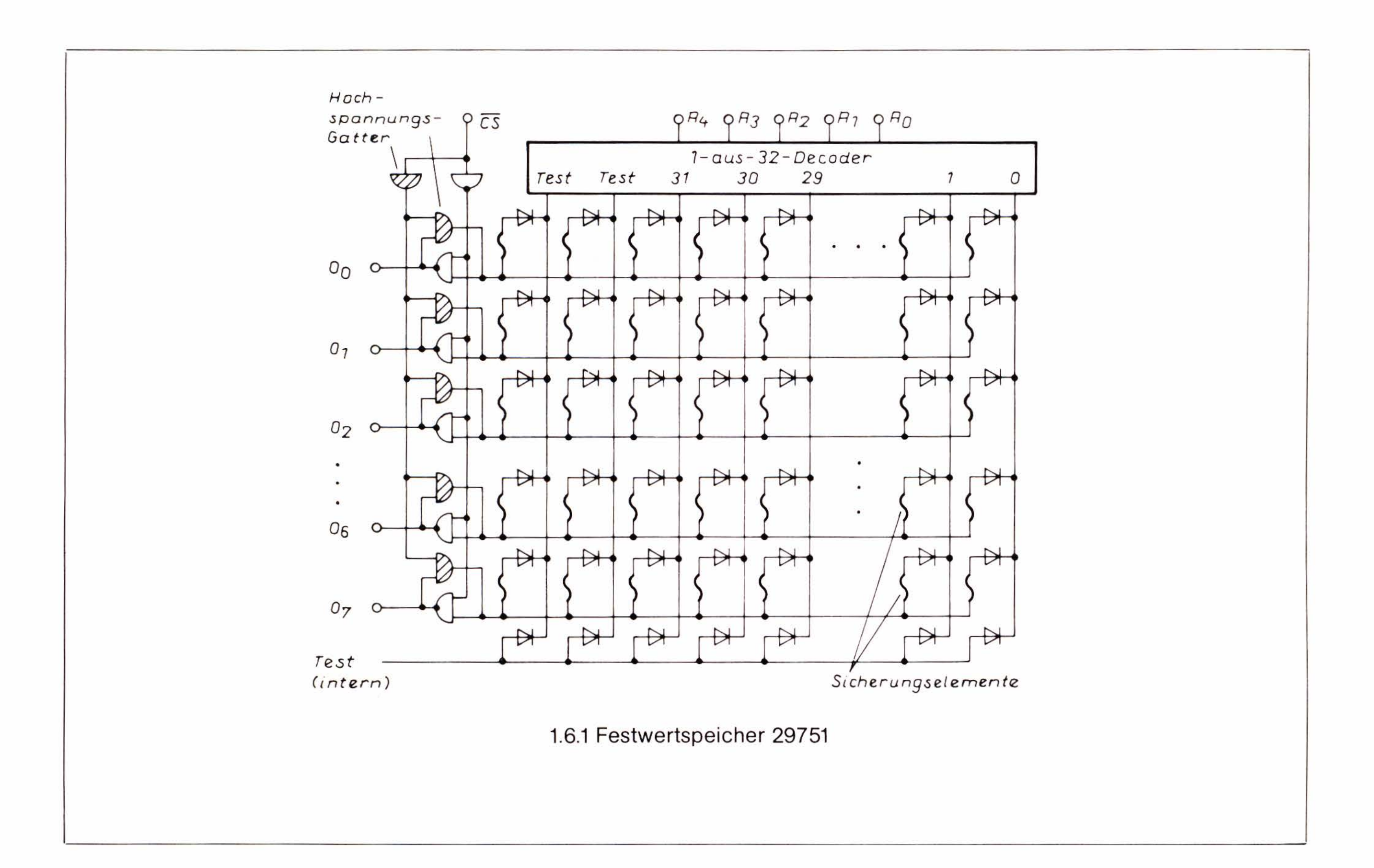
Tri-State-Ausgänge an den O-Punkten (Output). Das Auslesen einer Information ist zerstörungsfrei, d. h. die eingeschriebene Information wird beim Auslesen nicht verändert und kann daher beliebig oft verwendet werden. Hat der $\overline{CS}$-Eingang ein H-Signal, so sind die Ausgänge des RAMs wieder hochohmig.
Der Baustein 2703 hat eine Besonderheit. Während des Einschreibens einer Information stehen diese sofort an dem Ausgang zur Verfügung, da kein R/W-Steuereingang (Lese-/Schreib-Eingang) für einen getrennten Schreib- und Lese-Betrieb vorhanden ist.
In Bild 1.5.2 ist das Anschlußbild des Bausteines 2703 gezeigt.

1.6 Festwertspeicher 29751

Ein Festwertspeicher enthält festprogrammierte Daten, d. h. ein Festwertspeicher läßt sich nur auslesen. Ein Festwertspeicher wird auch als ROM (read only memory) bezeichnet.
In Bild 1.6.1 ist die Innenschaltung des ROMs 29751 gezeigt. Über die fünf Adresseneingänge A_0 bis A_4 werden die 32 Zeilen der Speicheranordnung angesteuert. Jede Diode in der Speicheranordnung bildet zusammen mit dem Sicherungselement eine Speicherzelle. Ist das Sicherungselement vorhanden, bezeichnet man die Speicherzelle als nicht programmiert. Durch einen Stromstoß wird das Sicherungselement in einer Speicherzelle zerstört und damit ist die Zelle programmiert.
Nach dem Programmieren des Bausteines läßt sich der Inhalt nicht mehr löschen. Liegt an dem Baustein eine Adresse, wird die betreffende Zeile auf die Ausgänge geschaltet. Da wir acht Ausgänge haben, besteht jede Zeile aus acht Dioden mit acht Sicherungselementen. Mit der Adresse können wir die Speicherzellen auf die Ausgänge schalten. Hat der $\overline{CS}$-Eingang ein H-Signal, so sind die Ausgangsbuffer des Bausteines hochohmig. Die einzelnen Speicherzellen können auf den Ausgang geschaltet werden, wenn an dem $\overline{CS}$-Eingang ein L-Signal liegt.
Das Programmieren eines Festwertspeichers ist ein schwieriges Kapitel. Normalerweise können wir einen Festwertspeicher einfach programmieren, wenn ein ROM-Programmer vorhanden ist, aber dieses Gerät kostet ca. 1000,– DM.
Eine andere, aber sehr umständliche Programmierung läßt sich selbst durchführen. Wir wollen auf der Adresse ADR 0 folgendes Wort programmieren:

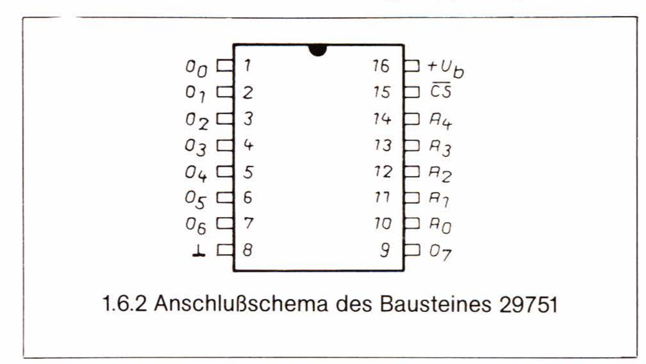
ADR	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7
0	L	H	L	L	L	L	L	L

Der Ausgang 0_1 soll ein H-Signal haben, wenn die Adresse 0 an dem Baustein anliegt. Der Decoder hat

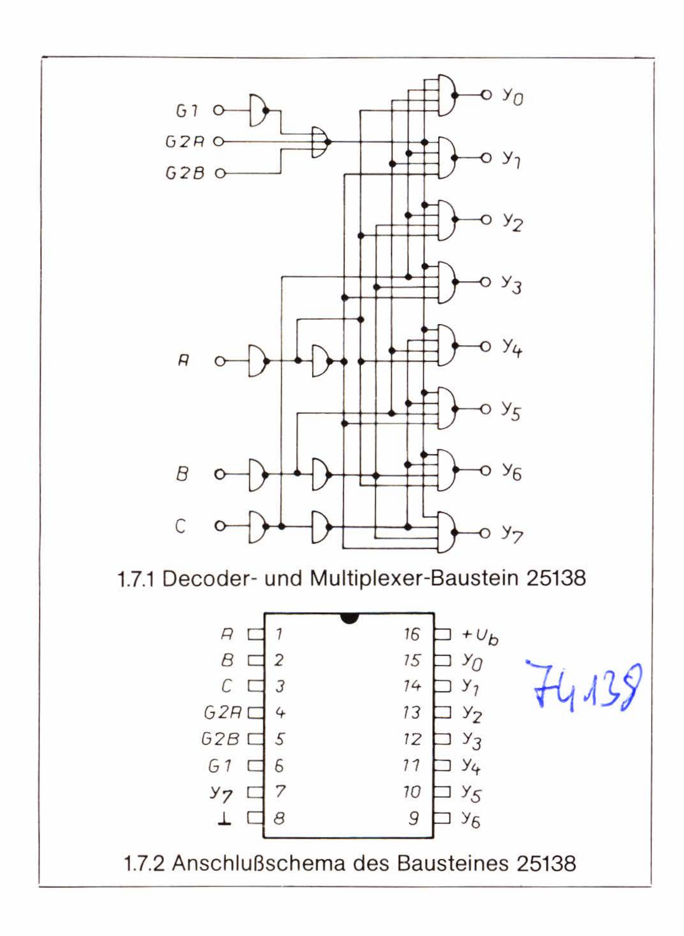


1.6.1 Festwertspeicher 29751

auf der Zeile 0 ein L-Signal, wenn die Adresse anliegt. Das Ausgangs-NAND-Gatter mit vorgeschaltetem Leseverstärker hat ein H-Signal. Ist das Sicherungselement vorhanden, so wird das H-Signal des Leseverstärkers zu L-Signal und der Ausgang des NAND-Gatters zu H-Signal. Ist das Sicherungselement nicht mehr vorhanden, so erkennt der Leseverstärker ein H-Signal und der Ausgang des NAND-Gatters hat ein L-Signal. Wir müssen auf der Adresse ADR 0 alle Sicherungselemente abbrennen, bis auf das Sicherungselement in der zweiten Spalte (Ausgang 0_1). Wir legen die Adresse ADR 0 an. Danach erhält der Ausgang 0_0 eine Spannung von 25 V, d. h. der Ausgang des Bausteines wird auf eine Betriebsspannung von 25 V hochgelegt. Mit einem kurzen Impuls an dem $\overline{CS}$-Eingang fließt ein Strom über das Hochspannungs-Gatter, dem Sicherungselement und der Diode nach Masse ab. Da das Sicherungselement sehr dünn ist, schmilzt es durch und die Speicherzelle ist programmiert. Mit dem Signal an dem $\overline{CS}$-Eingang steuern wir den zeitlichen Ablauf der Programmierung. Der Impuls muß genau 1 ms lang sein. Ist der Impuls zu kurz, brennt das Sicherungselement nicht exakt ab. Ist der Impuls zu lange, kann das Hochspannungs-Gatter beschädigt werden.
Nach der Programmierung dieser Speicherzelle legen wir die 25 V auf den Ausgang 0_2. Mit einem Programmierimpuls von 1 ms an dem $\overline{CS}$-Eingang schmilzt das nächste Sicherungselement durch. Danach wird die Spannung an den Ausgang 0_3 gelegt usw. Für uns heißt das, soll in dem Festwertspeicher ein L-Signal gespeichert werden, so müssen wir die betreffende Sicherung durchschießen (abbrennen). Bei einem H-Signal bleibt das Sicherungselement vorhanden.
Wenn wir die Adresse 0 programmiert haben, erhöhen wir die Adresse auf ADR 1 und beginnen wieder mit dem Programmiervorgang. Dieses Programmieren ist sehr umständlich und kann leicht zu Fehlern führen. Ist ein Sicherungselement durchgeschossen, so kann es nicht mehr repariert werden!!! Der Baustein kann weggeworfen werden.
Der Festwertspeicher in dem Mikrocomputer-Modell ist bereits programmiert. Bitte keine Experimente mit Programmierversuchen, da kein Ersatz gewährt wird. In Bild 1.6.2 ist die Anschlußbelegung gezeigt.



1.6.2 Anschlußschema des Bausteines 29751



1.7.1 Decoder- und Multiplexer-Baustein 25138

1.7.2 Anschlußschema des Bausteines 25138

1.7 Decoder und Demultiplexer 25138

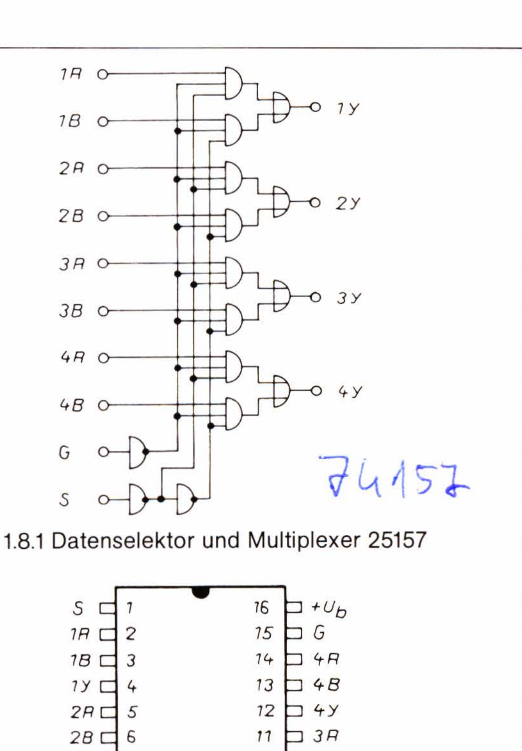
Der Baustein 25138 kann als Decoder und Demultiplexer verwendet werden. In Bild 1.7.1 ist die Innenschaltung für den Baustein gezeigt. Die Wertigkeiten liegen an den drei Eingängen A, B und C. Damit ergibt sich folgende Funktionstabelle:

Eingänge			Ausgang
C	B	A	
L	L	L	Y_0
L	L	H	Y_1
L	H	L	Y_2
L	H	H	Y_3
H	L	L	Y_4
H	L	H	Y_5
H	H	L	Y_6
H	H	H	Y_7

Mit den drei Eingängen erfolgt die Ansteuerung der acht NAND-Gatter. Die Eingangsdaten können an einen der drei Eingänge G1, G2A oder G2B angeschlossen werden. Welche Eingänge man verwendet, hängt von dem jeweiligen Anwendungsfall ab. Die beiden anderen Eingänge kann man als Sperr-Eingänge verwenden. Es ergibt sich folgende Funktionstabelle:

G1	G2A	G2B	Ausgangsfunktionen
L	L	L	Bei G1 als Daten-Eingang, kann der Baustein arbeiten.
L	L	H	Bei G1 als Daten-Eingang, kann der Baustein arbeiten.
L	H	L	Bei G1 als Daten-Eingang, kann der Baustein arbeiten.
L	H	H	Bei G1 als Daten-Eingang, kann der Baustein arbeiten.
H	L	L	Baustein gesperrt
H	L	H	G2B als Daten-Eingang
H	H	L	G2A als Daten-Eingang
H	H	H	Baustein gesperrt

Arbeitet der Baustein mit dem G2B-Eingang als Daten-Eingang, so wird der Baustein durch ein L-Signal gesperrt oder durch ein H-Signal geöffnet. Dies gilt auch wenn man den G2A-Eingang als Daten-Eingang verwendet.
Mittels den drei Steuereingängen A, B und C wird die Eingangsinformation auf einen der acht Ausgänge geschaltet. Der Baustein arbeitet in diesem Fall als Demultiplexer. Verwendet man die drei Steuereingänge A, B und C als Ansteuerung, so ergibt sich ein Decoder. In Bild 1.7.2 ist das Anschlußbild des Bausteines 25138 gezeigt.



1.8.1 Datenselektor und Multiplexer 25157

1.8.2 Anschlußschema des Bausteines 25157

1.8 Datenselektor und Multiplexer 25157

Das Bild 1.8.1 zeigt die Innenschaltung des Bausteines 25157. An den Eingängen A und B liegen die Eingangsinformationen. Mit dem Selektiereingang S wird bestimmt, ob die A- oder B-Eingänge mit den Ausgängen verbunden sind. Dadurch erhält der Baustein die Funktion eines Multiplexers. Es ergibt sich in Verbindung mit dem Strobe-Eingang S folgende Funktionstabelle:

G	S	Ausgang
H	X	gesperrt
L	L	A
L	H	B

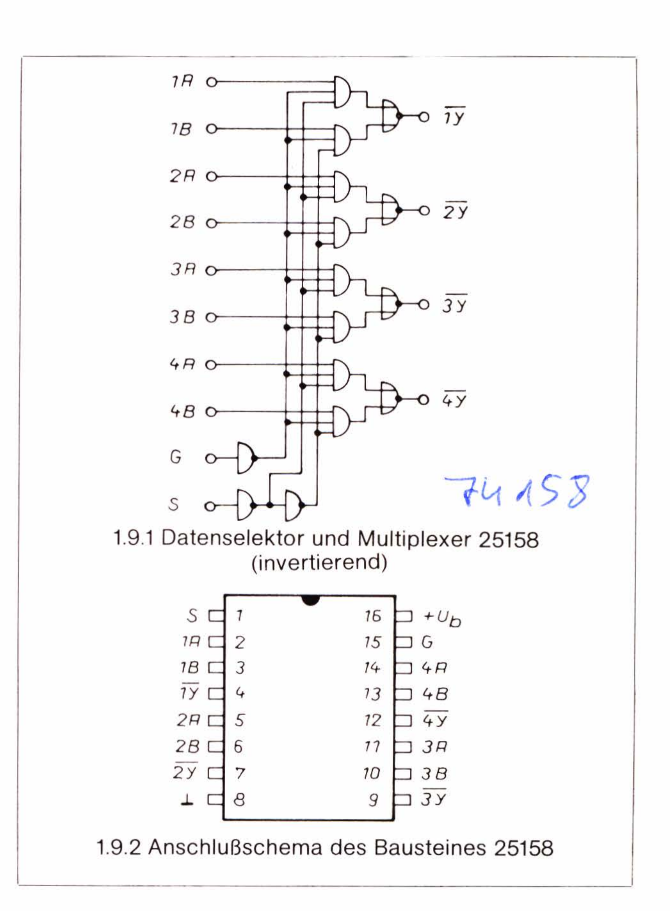
X = L- oder H-Signal

Wesentlich an dem Baustein 25157 ist die direkte Datenübertragung zwischen dem Eingang und dem Ausgang. Das Ausgangssignal liegt phasenrichtig mit dem Eingangssignal.

In Bild 1.8.2 ist das Anschlußbild des Bausteines 25157 gezeigt.

1.9 Datenselektor und Multiplexer 25158 (invertierend)

Der Baustein 25158 ist weitgehend mit dem Baustein 25157 identisch, aber die Ausgangsinformationen sind



1.9.1 Datenselektor und Multiplexer 25158 (invertierend)

1.9.2 Anschlußschema des Bausteines 25158

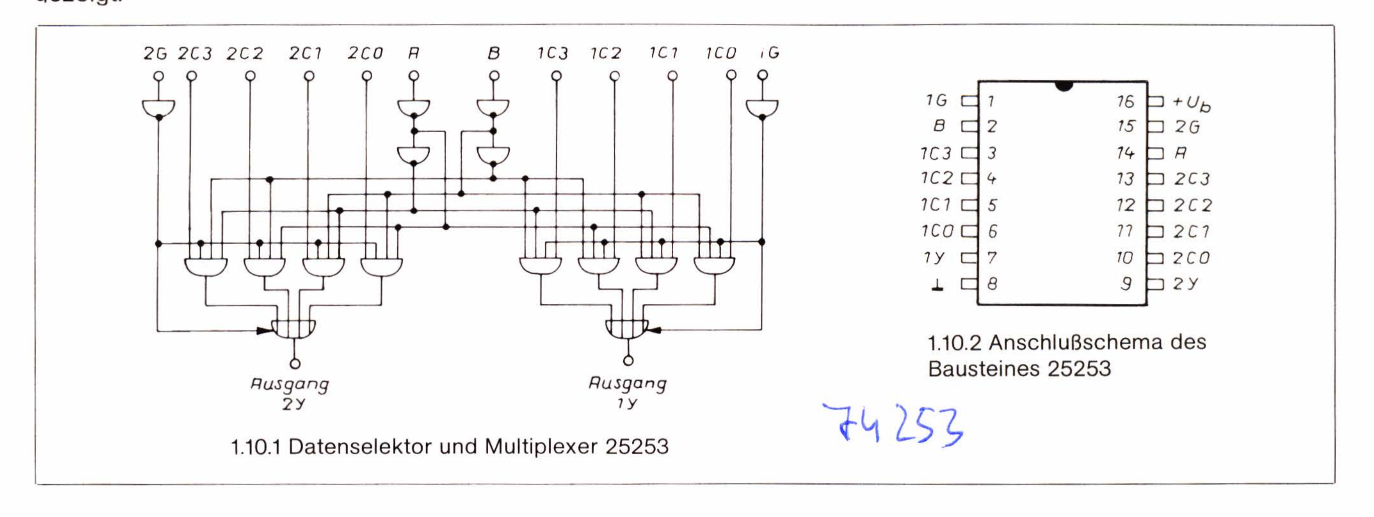
zu den Eingängen um 180° phasenverschoben. In Bild 1.9.1 ist die Innenschaltung des Bausteines gezeigt. Für den Baustein ergibt sich folgende Funktionstabelle:

G	S	Ausgang
H	X	gesperrt
L	L	$\overline{A}$
L	H	$\overline{B}$

In Bild 1.9.2 ist das Anschlußbild des Bausteines 25158 gezeigt.

1.10 Datenselektor und Multiplexer 25253

In Bild 1.10.1 ist die Innenschaltung des Bausteines 25253 gezeigt. Bei diesem Baustein werden jeweils vier Eingänge zu einem Ausgang zusammengefaßt. Die beiden Datenausgänge haben ein Tri-State-Verhalten.
Die Steuerung und die Ansteuerung des Bausteines wird durch die beiden Eingänge A, B und durch die beiden Eingänge 1G und 2G vorgenommen. Dadurch ergibt sich folgende Funktionstabelle:



1.10.1 Datenselektor und Multiplexer 25253

1.10.2 Anschlußschema des Bausteines 25253

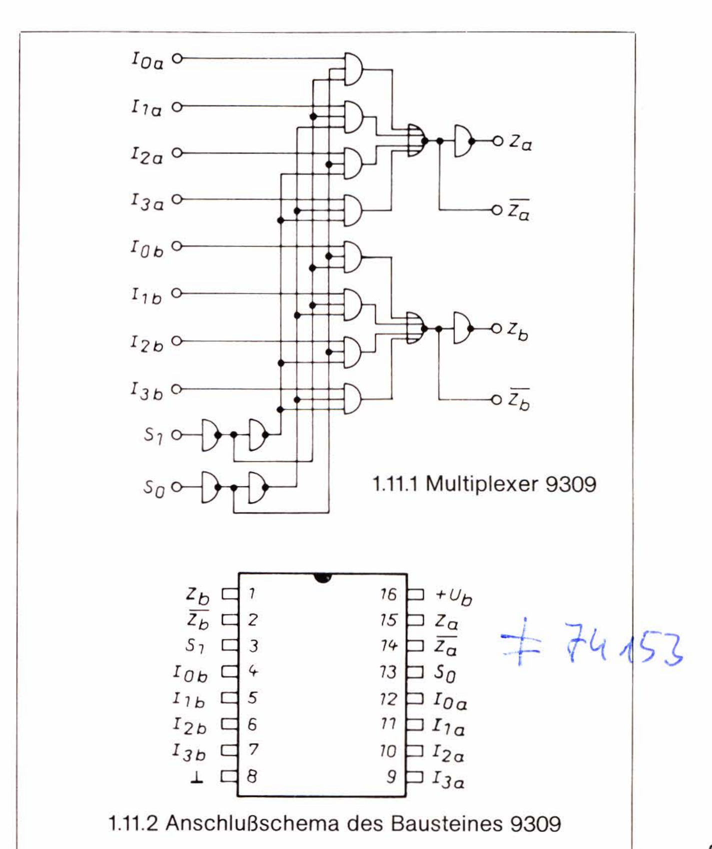
1G	2G	A	B	Ausgänge
H	H	X	X	1Y und 2Y sind hochohmig
L	L	L	L	C0-Eingänge
L	L	L	H	C1-Eingänge
L	L	H	L	C2-Eingänge
L	L	H	H	C3-Eingänge

Die beiden Eingänge 1G und 2G können voneinander separat angesteuert werden. Bei einem H-Signal ist das UND-Gatter gesperrt und der Tri-State-Ausgang wird hochohmig. Bild 1.10.2 zeigt das Anschlußschema des Bausteines 25253.

1.11 Multiplexer 9309

Im Bild 1.11.1 ist die Innenschaltung des Bausteines 9309 gezeigt. Der Baustein besitzt an seinen Ausgängen einen invertierten und einen nichtinvertierten Anschluß. Die Steuerung für den Baustein wird über die zwei S-Eingänge vorgenommen. Es ergibt sich folgende Funktionstabelle:

S_1	S_0	Ausgänge
L	L	I_0
L	H	I_1
H	L	I_2
H	H	I_3



1.11.1 Multiplexer 9309

1.11.2 Anschlußschema des Bausteines 9309

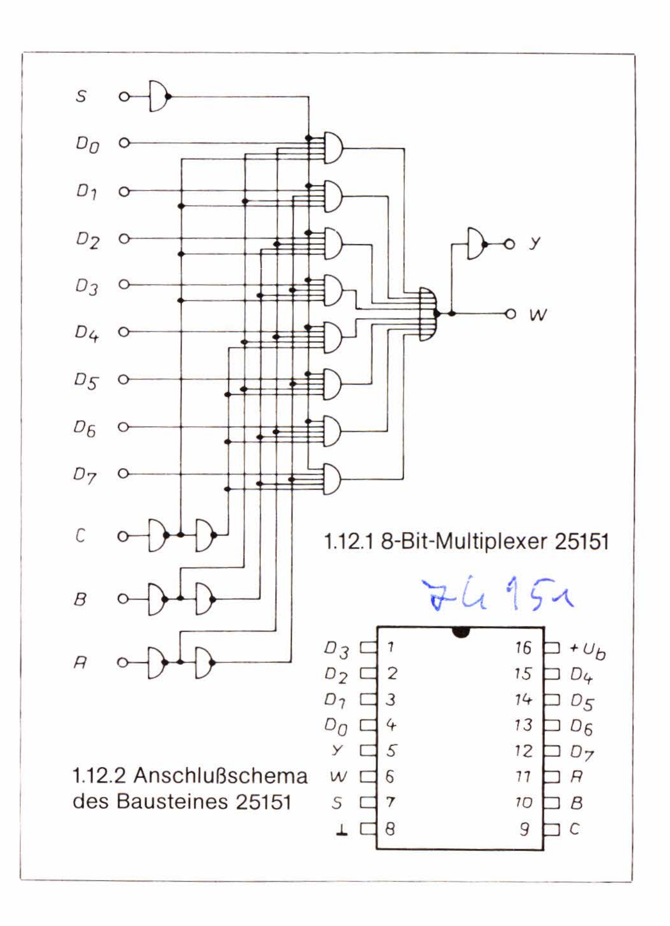
Eine Sperrung durch einen Enable-Eingang ist bei dem Baustein nicht vorhanden.
In Bild 1.11.2 ist das Anschlußbild für den Baustein 9309 gezeigt.

1.12 8-Bit-Multiplexer 25151

In Bild 1.12.1 ist die Innenschaltung des 8-Bit-Multiplexers gezeigt. Über die acht D-Eingänge kann jeweils ein Eingang auf die beiden Ausgänge W und Y geschaltet werden. Welcher Eingang durchgeschaltet werden soll, kann durch die drei Eingänge A, B und C bestimmt werden. In Verbindung mit dem Sperr-Eingang S ergibt sich folgende Funktionstabelle:

S	C	B	A	Ausgänge	
				gesperrt	
H	X	X	X	Y	W
L	L	L	L	D_0	$\bar{D}_0$
L	L	L	H	D_1	$\bar{D}_1$
L	L	H	L	D_2	$\bar{D}_2$
L	L	H	H	D_3	$\bar{D}_3$
L	H	L	L	D_4	$\bar{D}_4$
L	H	L	H	D_5	$\bar{D}_5$
L	H	H	L	D_6	$\bar{D}_6$
L	H	H	H	D_7	$\bar{D}_7$

Das Anschlußschema ist in Bild 1.12.2 gezeigt.

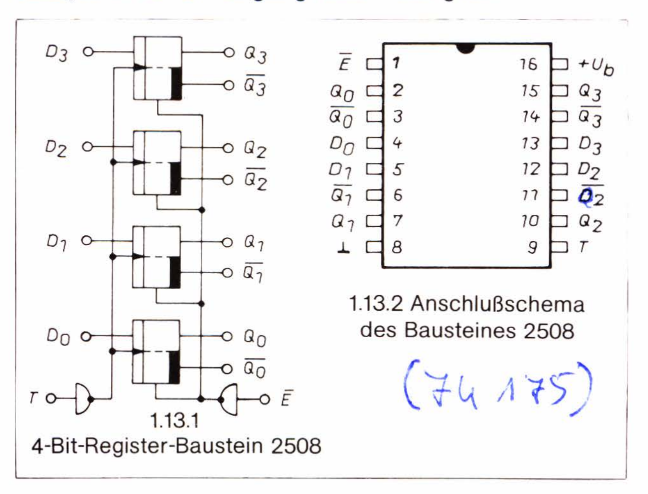


1.12.1 8-Bit-Multiplexer 25151

1.12.2 Anschlußschema des Bausteines 25151

1.13 4-Bit-Register-Baustein 2508

Die Innenschaltung für den Register-Baustein ist in Bild 1.13.1 gezeigt. Das Taktsignal liegt über einen invertierenden Buffer an den T-Eingängen der D-Flip-Flops. Mit einer positiven Taktflanke an dem Taktsignal-Eingang werden die an den D-Eingängen anstehenden Informationen in die einzelnen Flip-Flops übernommen. Mit einem L-Signal an den D-Eingängen wird ein Flip-Flop gesetzt, d. h. der Q-Ausgang hat ein H-Signal. Mit einem H-Signal wird dagegen das Flip-Flop zurückgesetzt, d. h. der Q-Ausgang hat ein L-Signal.



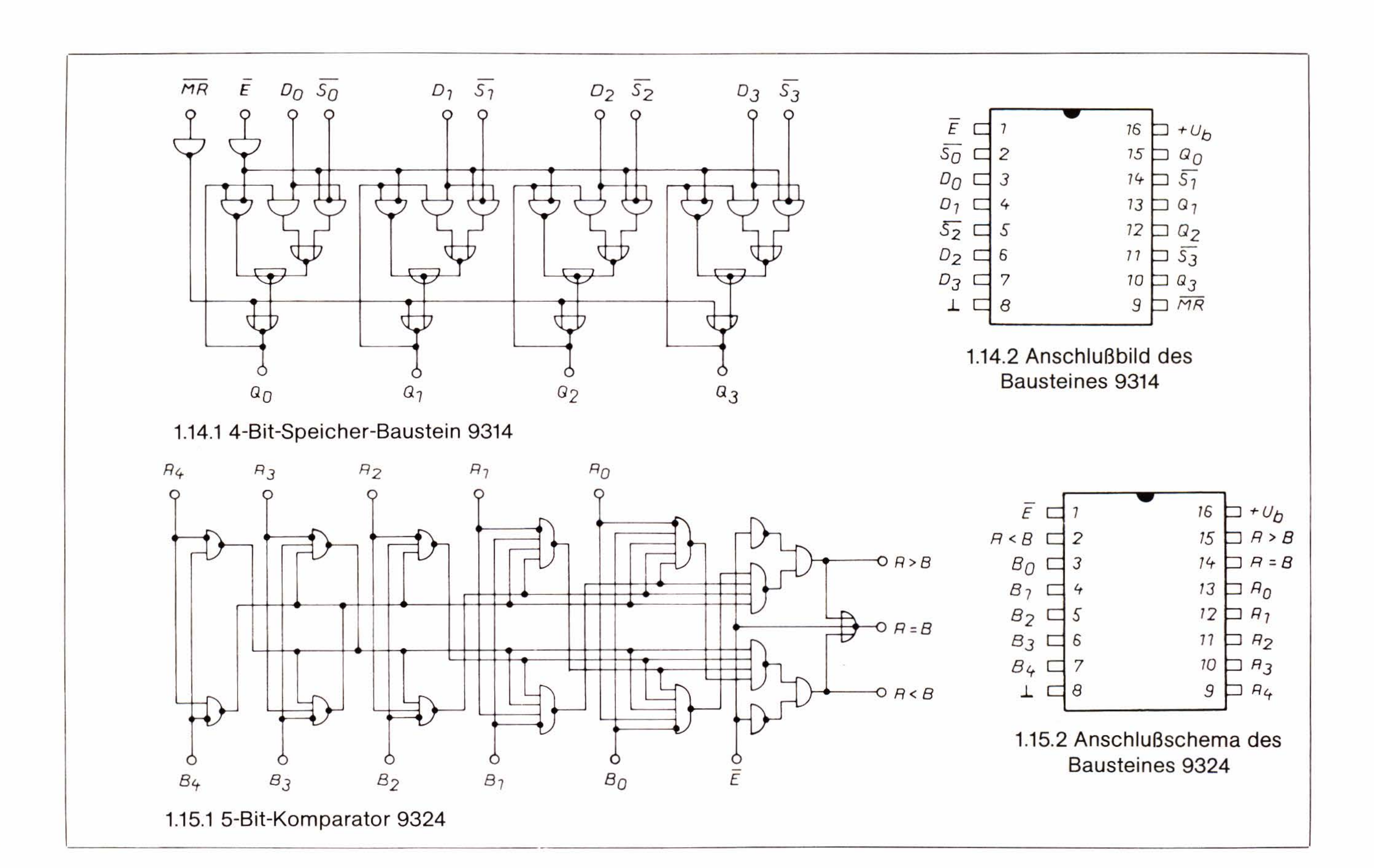
1.13.1 4-Bit-Register-Baustein 2508

1.13.2 Anschlußschema des Bausteines 2508

Der Enable-Eingang E hat eine besondere Funktion. Bei einem L-Signal werden die an den D-Eingängen anstehenden Informationen in die Flip-Flops übernommen, wenn ein positiver Taktimpuls erfolgt. Bei einem H-Signal an dem E-Eingang bleibt die Informationsaufnahme dagegen gesperrt. In Bild 1.13.2 ist das Anschlußschema für den Baustein 2508 gezeigt.

1.14 4-Bit-Speicher-Baustein 9314

Die Innenschaltung des Speicher-Bausteines 9314 ist in Bild 1.14.1 gezeigt. Bei den Speichern handelt es sich um D-Flip-Flops mit einem gemeinsamen Enable-Eingang $\overline{E}$ und einem gemeinsamen Master-RESET-Eingang $\overline{MR}$. Mit diesem $\overline{MR}$-Eingang kann der gesamte Speicherinhalt des Bausteines gelöscht werden, d. h. die Q-Ausgänge haben danach ein L-Signal.
Die Eingangsinformationen liegen an den D-Eingängen. Die Informationen werden gespeichert, wenn die $\overline{S}$-Eingänge von H- nach L-Signal schalten. Bei einem L-Signal erfolgt die Speicherung. Wichtig für einen Speichervorgang ist die Tatsache, daß immer nur die entgegengesetzte Information gespeichert werden kann, d. h. ist in einem D-Flip-Flop ein L-Signal gespeichert, so kann ein L-Signal an dem D-Eingang das Flip-Flop nicht kippen. Dies gilt auch für das H-Signal.
In Bild 1.14.2 ist das Anschlußschema des Bausteines 9314 gezeigt.



1.14.1 4-Bit-Speicher-Baustein 9314

1.14.2 Anschlußbild des Bausteines 9314

1.15.1 5-Bit-Komparator 9324

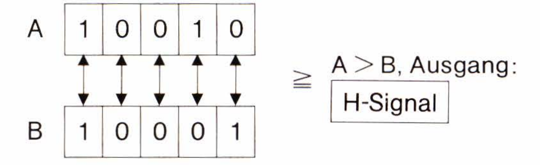
1.15.2 Anschlußschema des Bausteines 9324

1.15 5-Bit-Komparator 9324

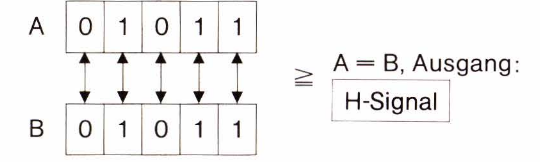
In Bild 1.15.1 ist die Innenschaltung für den 5-Bit-Komparator-Baustein 9324 gezeigt. Aus den zwei 5-Bit-Wörtern werden folgende Ausgangsfunktionen gebildet:

A > B
A = B
A < B

Beispiele für die einzelnen Funktionen:



Die beiden anderen Ausgänge haben ein L-Signal



Die beiden anderen Ausgänge haben ein L-Signal

A	0	0	1	1	1	≧ A < B, Ausgang: H-Signal
B	0	1	1	0	0	

Die beiden anderen Ausgänge haben ein L-Signal

Der Komparator vergleicht die einzelnen A- und B-Eingänge miteinander und bildet zwei Quersummen. Die beiden Quersummen ergeben den A > B-Ausgang und den A < B-Ausgang. Haben beide Ausgänge ein L-Signal, ist die NOR-Bedingung für den A = B-Ausgang erfüllt.
Mit einem H-Signal an dem $\overline{E}$-Eingang können die drei Ausgänge gesperrt werden, d. h. die drei Ausgänge haben ein L-Signal. Der Baustein kann nur arbeiten, wenn der E-Eingang auf L-Signal liegt.
In Bild 1.15.2 ist das Anschlußschema für den Komparator-Baustein 9324 gezeigt.

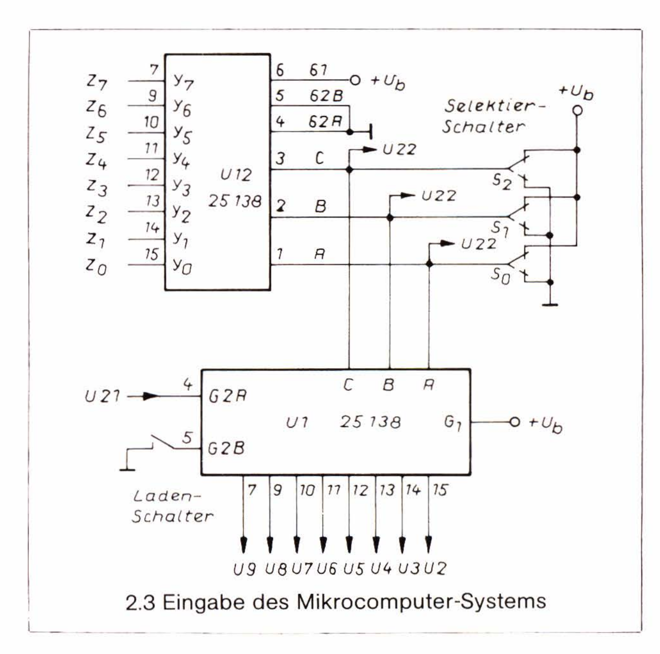
2. Zusammenschaltung des Mikrocomputer-Systems

In Bild 2.1 und 2.2 (als Anhang am Ende des Buches) ist die gesamte Zusammenschaltung für das Mikrocomputer-System gezeigt. In diesem Kapitel wird jede einzelne Funktion dieses Systems analysiert und im Detail erklärt. Aus dieser Hardware des Systems kann dann die Software einfach erstellt werden, da die einzelnen Funktionen weitgehend bekannt sind.

In Bild 2.3 ist der Selektier-Schalter in Verbindung mit dem zwei Decoder-Bausteinen vom Typ 25138 gezeigt. Die drei Selektier-Schalter sind mit dem U1 und U12 verbunden.

Der Baustein U1 steuert mit seinen Ausgängen die Schreib-Lese-Speicher des Systems an. Da das System acht Schreib-Lese-Speicher beinhaltet, benötigen wir drei Selektier-Variable, die durch den U1 in acht Ansteuerungs-Variable umgewandelt werden. Wir bestimmen mit den Selektier-Schaltern welchen Schreib-Lese-Speicher wir ansteuern.

An dem G2A-Eingang liegt der Ausgang von dem Baustein U21 an. Arbeitet der Mikroprozessor, so hat dieser Eingang ein H-Signal. Damit können die einzelnen Schreib-Lese-Speicher von den Selektier-Schaltern nicht angesteuert und der Programmfluß nicht unterbrochen werden. Ist ein Rechenprozeß beendet, meldet es der U21-Baustein mit einem L-Signal



2.3 Eingabe des Mikrocomputer-Systems

dem U1. Mit dem Laden-Schalter können wir nun den U1 aktivieren, d. h. wir können durch die Selektier-Schalter die einzelnen Schreib-Lese-Speicher ansteuern. Der G_1-Eingang ist mit $+U_b$ verbunden.

Mit dem U12-Baustein werden die Register-Bausteine 2918 angesteuert. In Bild 2.1 sind diese Bausteine mit den Nummern U13 bis U20 gekennzeichnet. Bei einer

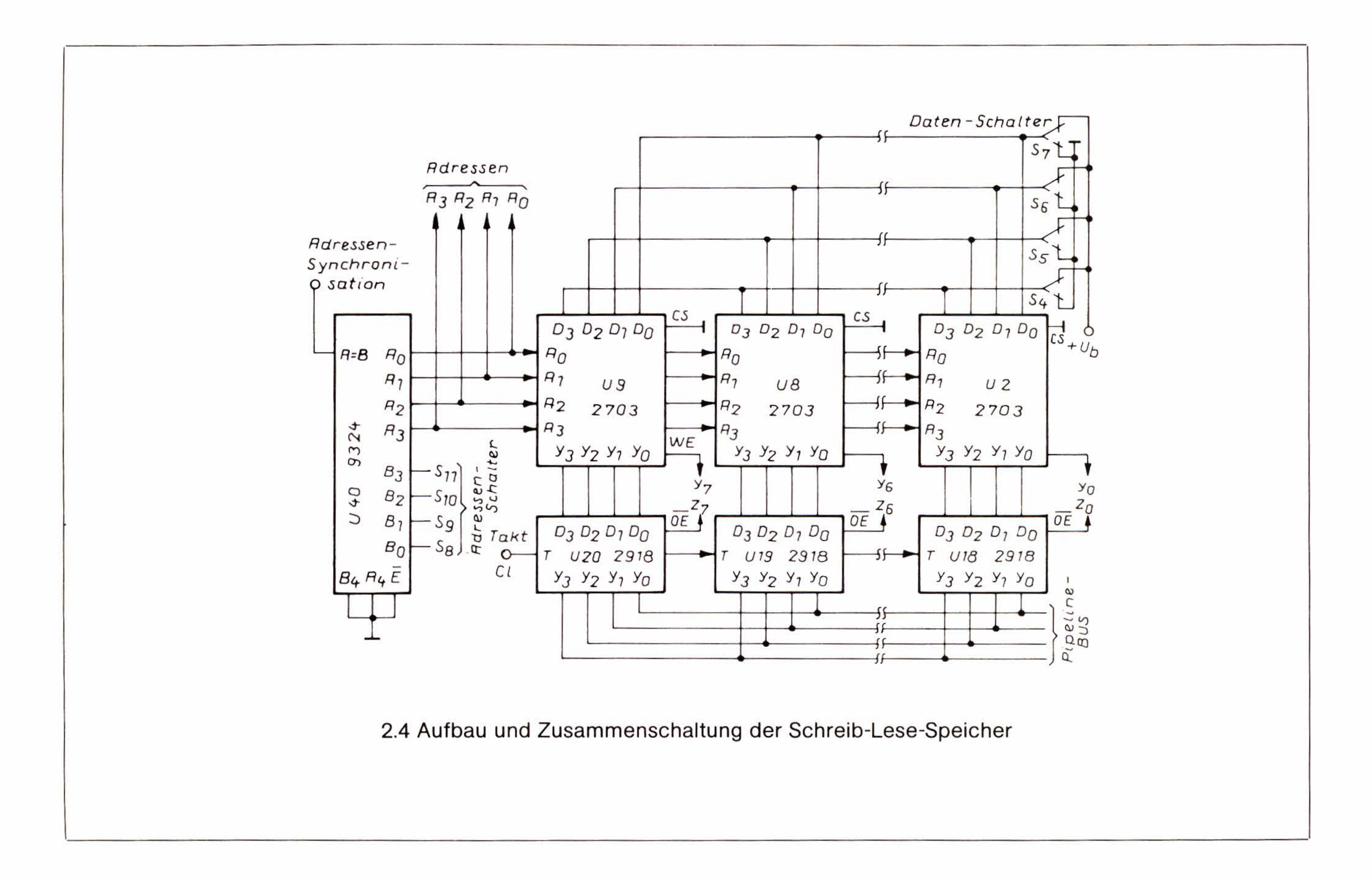
Selektion durch die drei Schalter hat ein bestimmter Ausgang des U12-Bausteines ein H-Signal. Mit diesem Signal wird einer der Register-Bausteine angesteuert und kann seine gespeicherten Informationen auf den Pipeline-BUS schalten. Damit sehen wir an den Pipeline-LEDs die gespeicherte Information. Die anderen Ausgänge des U12-Bausteines haben zu diesem Zeitpunkt ein L-Signal, d. h. die anderen Register-Baustein-Ausgänge sind hochohmig (Tri-State-Verhalten). Mit den Selektions-Schaltern können wir während des Rechenbetriebes die gespeicherten Funktionen in den einzelnen Speicherplätzen durch den Pipeline-BUS und die Pipeline-LEDs abfragen, d. h. sichtbar darstellen. Das Bild 2.4 zeigt den Aufbau des Schreib-Lese-Speichers mit den Daten-Schaltern, den Register-Bausteinen und dem Komparator. Durch den Mikroprogramm-Sequenzer erhalten die Schreib-Lese-Speicher und der Komparator die Adressen.
Wenn der Mikroprozessor zu arbeiten beginnt, schaltet er auf die Adressen ADR 0. Damit wird der gespeicherte Inhalt der Schreib-Lese-Speicher auf die Ausgänge geschaltet. Von den Ausgängen werden die einzelnen Funktionen im Mikrocomputer-System abgerufen. Sind die einzelnen Befehle durchgeführt worden, erhöht der Mikroprozessor durch den Mikroprogramm-Sequenzer seine Adresse um 1. Damit werden die nächsten Daten und Befehle auf die Ausgänge der Schreib-Lese-Speicher geschaltet. Durch die 16 Adressen des Mikrocomputer-Systems ergeben sich 16 Befehls- und Datenebenen.
Die einzelnen Daten und Befehle geben wir über die Daten-Schalter im Dual-Code ein. Die Daten und Befehle werden im einzelnen beschrieben. Durch die Adressen steuern wir die einzelnen Adressen-Ebenen des Mikrocomputer-Systems an. Damit können wir in einer Ebene alle acht Schreib-Lese-Speicher gleichzeitig adressieren. Mit den Selektier-Schaltern bestimmen wir, in welches RAM wir Daten oder Befehle einschreiben. Die Daten-Eingabe und Adressierung erfolgt im Dual-Code.

Funktionen der einzelnen Schreib-Lese-Speicher:

U2 (L L L): In diesem Schreib-Lese-Speicher werden die Daten gespeichert. Der Daten-Bereich liegt zwischen 0 und 15.

U3 (L L H): In diesem Schreib-Lese-Speicher werden die Adressen für den internen Schreib-Lese-Speicher des Mikroprozessors gespeichert. Der Adressen-Bereich liegt zwischen 0 und 15.

U4 (L H L): In diesem Schreib-Lese-Speicher werden die Adressen für den internen Schreib-Lese-Speicher des Mikroprozessors gespeichert. Der Adressen-Bereich liegt zwischen 0 und 15.



2.4 Aufbau und Zusammenschaltung der Schreib-Lese-Speicher

U5 (L H H): In diesem Schreib-Lese-Speicher werden die Funktionen der ALU gespeichert. Es ergibt sich folgende Struktur:

Nr.	C_n	Funktion
0	X	$R + S$
1	X	$S - R$
2	X	$R - S$
3	X	$R \vee S$
4	X	$R \wedge S$
5	X	$\overline{R} \wedge S$
6	X	$R \veebar S$
7	X	$\overline{R \veebar S}$

Wir können insgesamt acht Funktionen für die ALU eingeben. Dazu sind die drei Schalter S_5, S_6 und S_7 vorhanden. Mit dem Schalter S_4 geben wir den Übertrag C_n ein. Mit einem 1-Signal (H-Potential) erhält die betreffende Rechenoperation einen Übertrag, der bestimmte Funktionen auslösen kann, wie in den Programmier-Beispielen noch ausführlich gezeigt wird. Wird kein Übertrag benötigt, geben wir ein 0-Signal (L-Potential) ein.

U6 (H L L): In diesem Schreib-Lese-Speicher werden die Funktionen für die ALU-Eingänge gespeichert. Es ergibt sich folgende Struktur:

		Funktion	
Nr.	MUX_0	R	S
0	–	A	Q
1	–	A	B
2	–	Ø	Q
3	–	Ø	B
4	–	Ø	A
5	–	D	A
6	–	D	Q
7	–	D	Ø

Die Funktion des MUX_0 wird anschließend in Verbindung mit dem MUX_1 noch beschrieben. Wir können durch die einzelnen Kombinationen bestimmen, welcher Daten-Eingang an der ALU anliegt. Bei Nr. 0 werden z. B. die Daten von A mit den Daten von Q kombiniert. Welche Rechenoperation durchgeführt werden soll, bestimmen wir durch den Schreib-Lese-Speicher U5 mit den ALU-Funktionen.
Bei den Funktionen kann ein Eingang mit einer Ø kombiniert werden, d. h. bei einer Addition mit einer 0 bleibt der eingegebene Summand erhalten und es kommt nur zu einer Übertragungsfunktion.

U8 (H L H): In diesem Schreib-Lese-Speicher werden die Funktionen für den einzelnen Lade-Betrieb gespeichert. Es ergibt sich folgende Struktur:

Nr.	MUX_1	Lade-Funktion	Y
0	–	$F \rightarrow Q$	F
1	–	keine	F
2	–	$F \rightarrow B$	A
3	–	$F \rightarrow B$	F
4	–	$F/2 \rightarrow B$ $Q/2 \rightarrow Q$	F
5	–	$F/2 \rightarrow B$	F
6	–	$2F \rightarrow B$ $2Q \rightarrow Q$	F
7	–	$2F \rightarrow B$	F

Mit der Funktion 0 bestimmen wir z. B., daß das Ergebnis einer Rechenoperation von der ALU aus in das Q-Register abgespeichert wird. Mit den einzelnen Ladefunktionen können wir das Ergebnis in eine der Speicher einschreiben und sie dann später wieder auslesen.

Für den MUX_1- und MUX_0-Betrieb ergibt sich folgende Tabelle:

MUX_1	MUX_0	Funktion	Abwärts		Aufwärts	
0	0	Nullzeichen	$0 \rightarrow RAM_3$	$0 \rightarrow Q_3$	$0 \rightarrow RAM_0$	$0 \rightarrow Q_0$
0	1	Rotieren	$RAM_0 \rightarrow RAM_3$	$Q_0 \rightarrow Q_3$	$RAM_3 \rightarrow RAM_0$	$Q_3 \rightarrow Q_0$
1	0	Doppel-Rotieren	$RAM_0 \rightarrow Q_3$	$Q_0 \rightarrow RAM_3$	$RAM_3 \rightarrow Q_0$	$Q_3 \rightarrow RAM_0$
1	1	Doppel-Arithmetik	$F_3 \rightarrow RAM_3$	$RAM_0 \rightarrow Q_3$	$Q_3 \rightarrow RAM_0$	$0 \rightarrow Q_0$

Mit den beiden MUX können wir die RAM_0-, RAM_3-, Q_0- und Q_3-Eingänge des Mikroprozessors beeinflussen. Dadurch können Multiplikationen und Divisionen durchgeführt werden. Dies gilt auch für spezielle Schiebefunktionen.

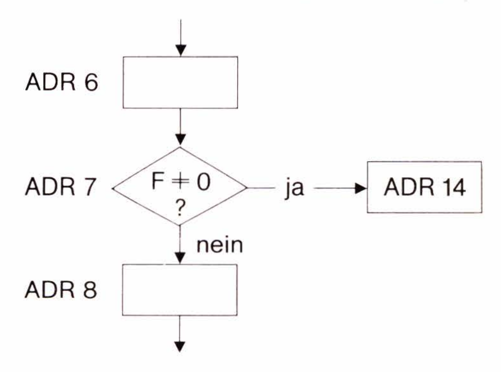
U7 (H H L): In diesem Schreib-Lese-Speicher geben wir die Befehle für die Mikroprozessor-Kontrolle ein. Mit diesen Befehlen können wir bedingte und unbedingte Programmsprünge durchführen. Es ergeben sich folgende Funktionen:

Codewort	Funktion
0	Zweig-Register, wenn $F \neq 0$
1	Zweig-Register
2	weiter
3	Zweig-Schalter (Daten-Schalter)
4	springe ins Unterprogramm, wenn $F \neq 0$
5	springe ins Unterprogramm
6	springe aus dem Unterprogramm
7	Stapelvergleich
8	Ende der Programmschleife und POP, wenn $F = 0$
9	PUSH und weiter
10	POP und weiter
11	Ende der Programmschleife und POP, wenn C_{n+4}
12	Zweig-Register, wenn $F = 0$
13	Zweig-Register, wenn F_3
14	Zweig-Register, wenn OVR
15	Zweig-Register, wenn C_{n+4}

Durch diese 16 Funktionen können wir mit diesem Mikrocomputer-System sämtliche gängigen Programmsprünge durchführen.

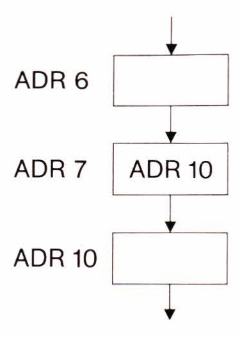
Für die einzelnen Codewörter ergeben sich folgende Programmsprünge:

Codewort 0: Der Mikroprozessor verläßt sein Programm, wenn $F \neq 0$. Damit wird eine Entscheidung im Programm getroffen, d. h. es entsteht ein bedingter Programmsprung. Das nachfolgende Flußdiagramm zeigt den Ablauf für einen bedingten Sprung:



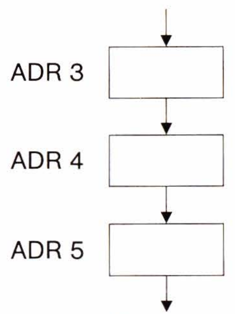
Auf der Adresse ADR 7 wird eine Entscheidung getroffen ... Hat der Ausgang $F \neq 0$ des Mikroprozessors **ein** L-(0-)Signal, springt das Programm auf die Adresse ADR 14. Hat der Ausgang $F \neq 0$ dagegen ein H-(1)-Signal, so wird das Programm auf der ADR 8 fortgesetzt. Mit der Entscheidung wird das Programm damit entweder auf der Adresse ADR 8 oder ADR 14 fortgesetzt. Die nächste Adresse hängt also von dieser Entscheidung ab und wir bezeichnen diesen Programmsprung daher als bedingten Sprung.

Codewort 1: Bei diesem Codewort ist der Mikroprozessor an keine Entscheidung gebunden. Tritt in einem Programmablauf dieses Codewort auf, so verläßt der Mikroprozessor sein Programm und springt auf eine Adresse, wie das nachfolgende Flußdiagramm zeigt:



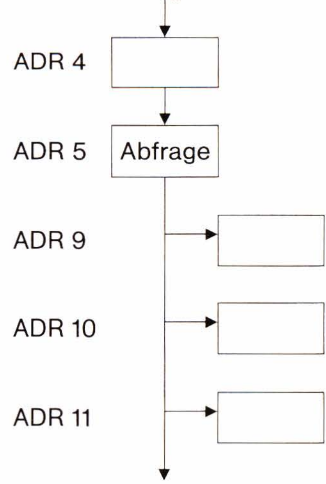
Der Mikroprozessor springt direkt von der Adresse ADR 7 auf die ADR 10. Das Codewort 1 ist damit ein unbedingter Sprungbefehl, d. h. der Programmablauf muß auf der ADR 10 fortgesetzt werden.

Codewort 2: Bei diesem Codewort haben wir ebenfalls einen unbedingten Sprung. Das Codewort 2 erhöht automatisch den Zählerstand des Mikroprogramm-Sequenzers, wenn der Befehl auf der Adresse ausgeführt worden ist, wie das nachfolgende Flußdiagramm zeigt:



Das Programm des Mikrocomputer-Systems wird bei diesem Befehl immer auf der nächsten Adresse weitergeführt.

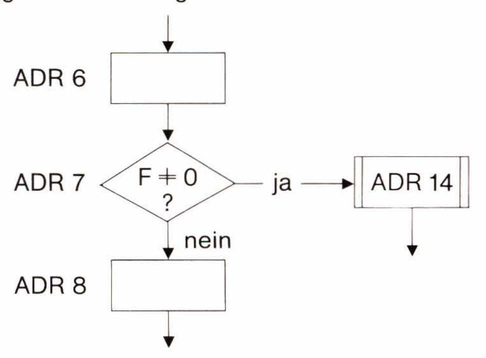
Codewort 3: Dieses Codewort stellt eine Besonderheit dar, wie das nachfolgende Flußdiagramm zeigt:



Bei der Adresse ADR 5 stoppt der Mikroprozessor. Mit den D-Schaltern können wir nun die nächste Adresse bestimmen. Sie kann zwischen 6 und 15 liegen.

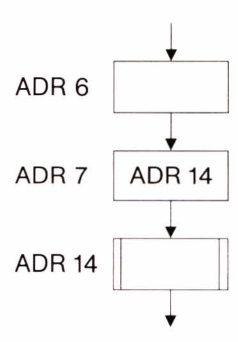
Wir stellen die nächste Adresse mit den D-Schaltern im Dual-Code ein. Erreicht der Mikroprozessor die Adresse ADR 5, so fragt der Mikroprozessor die D-Schalter ab und führt auf dieser externen Adressensteuerung sein Programm weiter. Wir haben hierbei weder einen bedingten noch unbedingten Sprung, sondern einen freiprogrammierbaren Programmsprung.

Codewort 4: Dieses Codewort ist ein bedingter Unterprogrammsprung. Der Mikroprozessor springt in sein Unterprogramm, wenn der Ausgang $F \neq 0$ des Mikroprozessors ein L-(0-)Signal hat. Das Unterprogramm befindet sich z. B. auf der Adresse ADR 14. Ist die Entscheidung anders, wird das Programm auf der nächsten Adresse des Programms weiter durchgeführt. Wir erhalten folgendes Flußdiagramm:



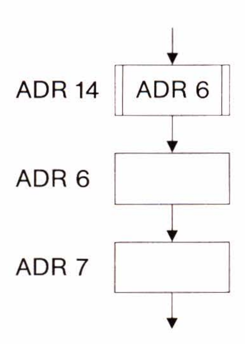
Durch die Entscheidung auf der Adresse ADR 7 erhalten wir einen bedingten Sprung.

Codewort 5: Bei diesem Codewort verläßt der Mikroprozessor unbedingt sein Programm und springt in das Unterprogramm, wie das folgende Flußdiagramm zeigt:

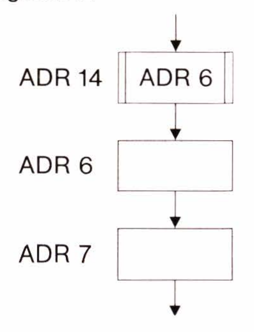


Der Mikroprozessor ist durch diesen unbedingten Sprung gezwungen, von der Adresse ADR 7 auf die ADR 14 in das Unterprogramm zu springen. Der Sprungbefehl ist von keiner Entscheidung abhängig.

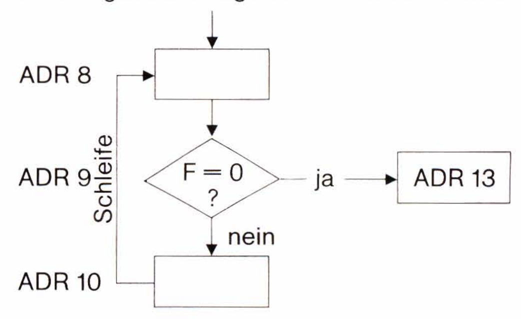
Codewort 6: Mit diesem Codewort verläßt der Mikroprozessor sein Unterprogramm und springt in sein Hauptprogramm wieder zurück. Es ergibt sich folgendes Flußdiagramm:



Es handelt sich hierbei um einen unbedingten Sprung.
Codewort 7: Mit diesem Codewort holt sich der Mikroprozessor aus seinem Stack-Pointer und Kellerspeicher seine letzte Adresse vor dem Sprung in das Unterprogramm. Damit kann der Mikroprozessor aus der letzten Adresse weiterarbeiten. Es ergibt sich folgendes Flußdiagramm:

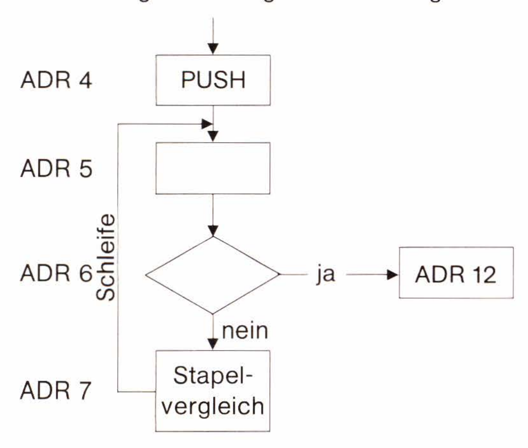


Der Rücksprung auf das Hauptprogramm ist ein unbedingter Sprung.
Codewort 8: Dieses Codewort können wir nur verwenden, wenn wir uns in einem Schleifenprogramm befinden. Es ergibt sich folgende Funktionstabelle:



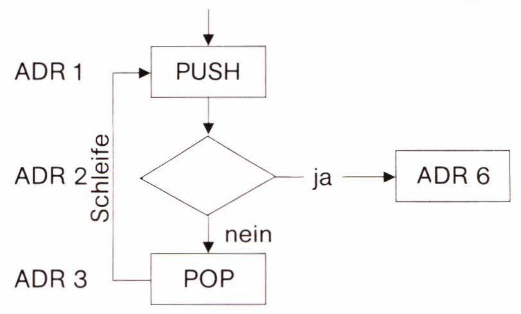
Nach der Adresse ADR 8 wird in dem Programm eine Entscheidung getroffen. Heißt die Antwort „nein" wird der Zählerstand um eine Eins auf die Adresse ADR 10 erhöht. Von der Adresse ADR 10 erfolgt der Rücksprung auf ADR 8. Der Mikroprozessor durchläuft die Schleife so lange, bis die Antwort „ja" heißt. Der Mikroprozessor steigt jetzt aus der Schleife aus und setzt sein Programm auf der Adresse ADR 13 fort. Wir haben einen bedingten Sprung.
Codewort 9: Mit dem Befehl „PUSH" wird die Adresse in dem Stapel-Speicher festgehalten. Befindet sich der Mikroprozessor innerhalb eines Schleifenprogrammes,

so kehrt das Adressen-Register in dem Mikroprogramm-Sequenzer immer auf diese festgehaltene Adresse zurück. Es ergibt sich folgendes Flußdiagramm:



Mit dem PUSH-Befehl leiten wir ein Schleifenprogramm ein. Auf der Adresse ADR 5 führt der Mikroprozessor eine arithmetische oder logische Funktion durch. Auf der Adresse ADR 6 erfolgt eine Entscheidung. Bei einem „nein" erfolgt ein Stapelvergleich und der Mikroprozessor kehrt aus seinem PUSH, also auf seinen Schleifenanfang zurück. Mit jeder Schleife wird eine Entscheidung in dem Programm getroffen. Ergibt sich auf der Adresse ADR 6 ein „ja", so verläßt der Mikroprozessor die Schleife und führt sein Programm auf der Adresse ADR 12 weiter. Wir haben einen bedingten Sprung aus einer Schleife.

Codewort 10: Der POP-Befehl ist ein unbedingter Rücksprung auf den PUSH. In dem folgenden Flußdiagramm ist der Aufbau für dieses Codewort gezeigt.

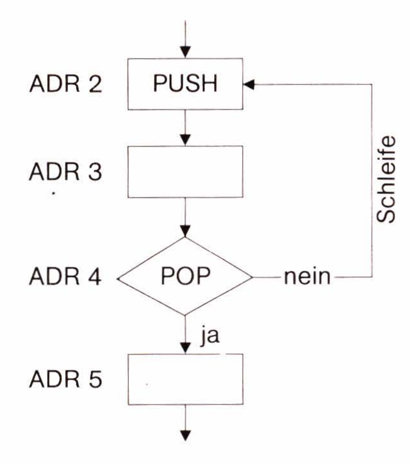


Mit einem PUSH-Befehl wird ein Schleifenprogramm eingeleitet. Der POP-Befehl beinhaltet den unbedingten Rücksprung auf den PUSH.

Der wesentliche Unterschied zwischen dem POP- und Stapelvergleich-Befehl ist sehr einfach. Bei einem Stapelvergleich kann auf dieser Adresse keine arithmetische oder logische Verknüpfung zusätzlich durchgeführt werden. Bei einem POP-Befehl können wir auf der gleichen Adresse nach einem arithmetischen oder logischen Befehl programmieren. Bei diesem Mikrocomputer-System wird zuerst der arithmetische oder logische Befehl ausgeführt und danach springt der

Mikroprozessor auf PUSH zurück. Wir sparen uns bei dem POP-Befehl eine Adresse.

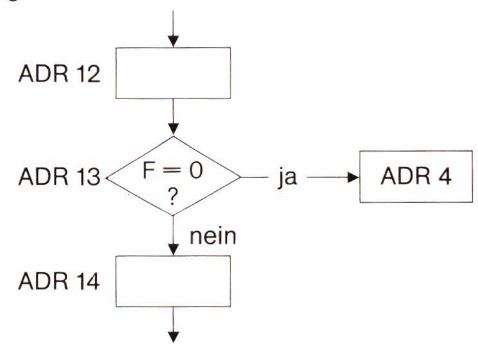
Codewort 11: Mit dem PUSH wird eine Programmschleife durch den Mikroprozessor eingeleitet. Der POP-Befehl stellt hier eine Besonderheit dar. Ist die Entscheidung positiv, so verläßt der Mikroprozessor automatisch sein Schleifenprogramm. Ist die Entscheidung negativ, springt er auf PUSH zurück und beginnt erneut das Schleifenprogramm. Es entsteht folgender Ablauf in dem Flußdiagramm:



Bei der Adresse ADR 4 wird die Entscheidung getroffen, aber mit POP. Bei einer negativen Entscheidung erfolgt der Rücksprung und bei einer positiven die Erhöhung der Adresse auf ADR 5.

Bei dem Codewort haben wir einen bedingten und einen unbedingten Sprung. Dies hängt von der Entscheidung ab. Da die Entscheidung relativ ist, sprechen wir von einem Relativ-Sprung. Ein Relativ-Sprung beinhaltet einen bedingten und einen unbedingten Sprung. Der Relativ-Sprung kann nur durchgeführt werden, wenn am Anfang einer Schleife ein PUSH-Befehl steht.

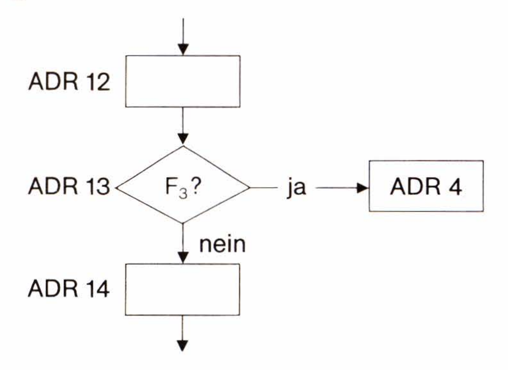
Codewort 12: Bei diesem Codewort führt der Mikroprozessor einen bedingten Sprung aus. Wir erhalten daher folgenden Ablauf:



Zwischen der Adresse ADR 12 und ADR 14 befindet sich eine Entscheidung. Ist die Entscheidung negativ, führt der Mikroprozessor sein Programm auf der Adresse ADR 14 weiter. Ist die Entscheidung dagegen

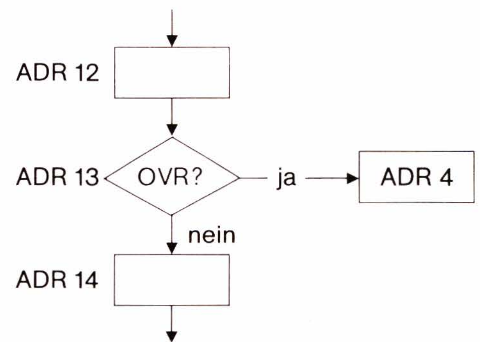
positiv, holt sich der Mikroprozessor eine Zweigadresse und springt auf diese. Wir haben also einen bedingten Sprung.

Codewort 13: Bei diesem Codewort führt der Mikroprozessor einen bedingten Sprung aus. Wir erhalten daher folgenden Ablauf:



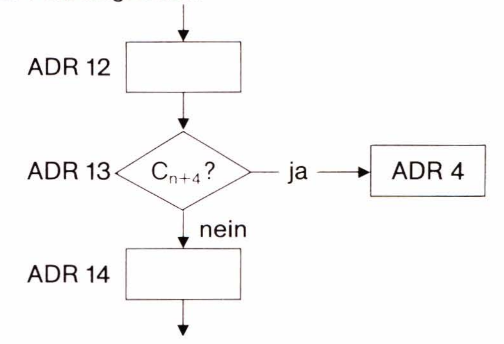
Auf der Adresse ADR 13 wird eine Entscheidung getroffen. Ist die Entscheidung negativ, führt der Mikroprozessor sein Programm auf der Adresse ADR 14 weiter. Ist die Entscheidung dagegen positiv, holt sich der Mikroprozessor seine Zweigadresse und springt auf diese. Wir haben also einen bedingten Sprung.

Codewort 14: Bei diesem Codewort führt der Mikroprozessor einen bedingten Sprung aus. Wir erhalten daher folgenden Ablauf:



Auf der Adresse ADR 13 trifft der Mikroprozessor eine Entscheidung. Ist die Entscheidung negativ, so führt der Mikroprozessor sein Programm auf der Adresse ADR 14 weiter. Ist die Entscheidung dagegen positiv, holt sich der Mikroprozessor seine Zweigadresse und springt auf diese. Wir haben einen bedingten Sprung.

Codewort 15: Der Mikroprozessor führt bei diesem Codewort einen bedingten Sprung aus und wir erhalten folgendes Flußdiagramm:



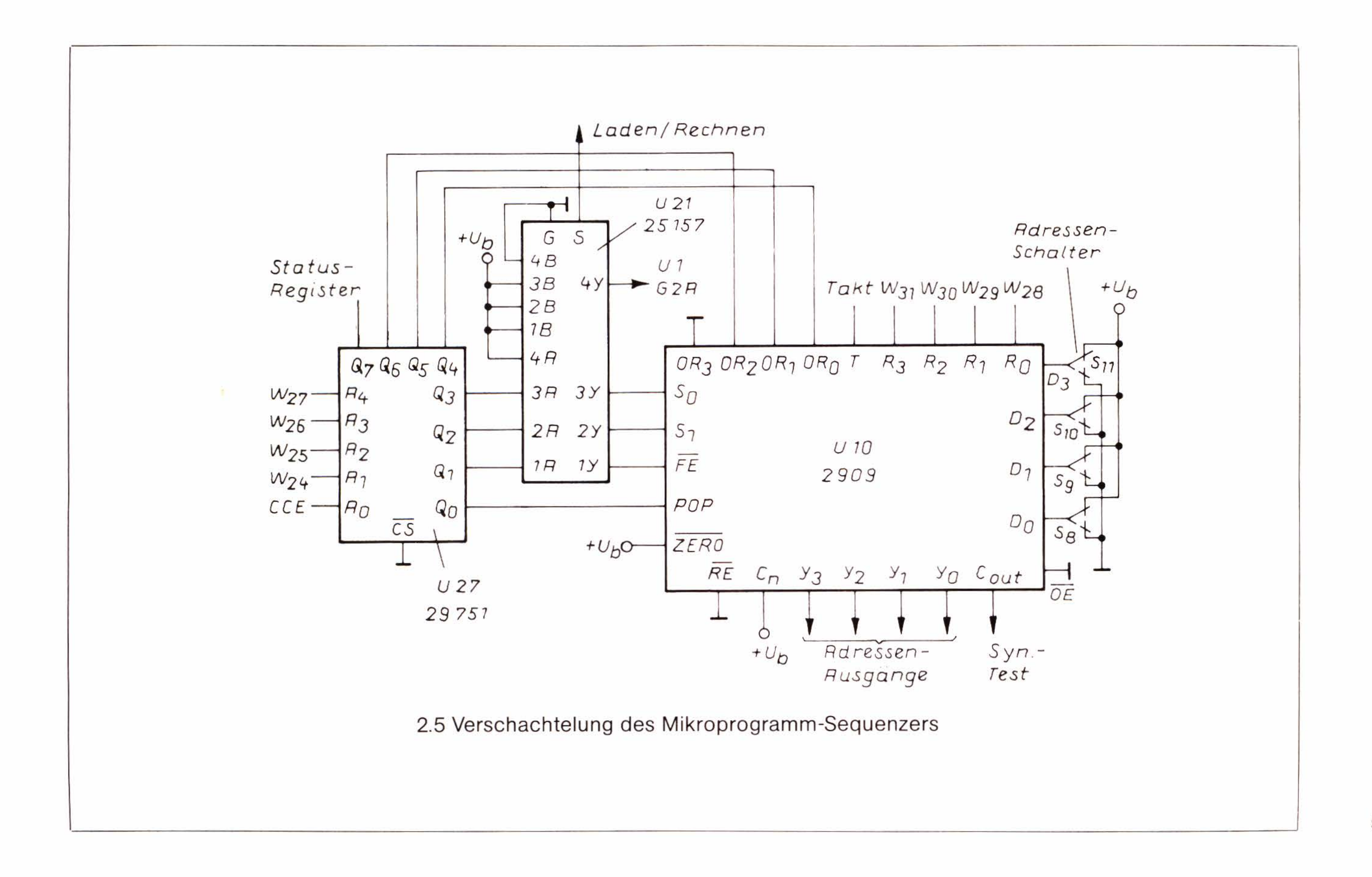
Auf der Adresse ADR 13 wird eine Entscheidung von dem Mikroprozessor getroffen. Ist die Entscheidung negativ, so führt der Mikroprozessor sein Programm auf der Adresse ADR 14 weiter. Bei einer positiven Entscheidung holt sich der Mikroprozessor seine Zweigadresse und springt auf diese. Wir haben damit einen bedingten Sprung.

U9 (H H H): In diesem RAM wird die Zweig-Adresse gespeichert. Damit kennt der Mikroprozessor seine nächste Adresse.

Das Bild 2.5 zeigt die Verschaltung des Mikroprogramm-Sequenzers mit dem Baustein 25157, dem Festwertspeicher 29751 und den Adressen-Schaltern.
Der Adressen-Schalter liegt an den D-Eingängen des Mikroprogramm-Sequenzers. Mit den Schaltern können wir die Adresse beim Programmieren des Mikrocomputer-Systems bestimmen. Um die Adressen auf die Y-Ausgänge schalten zu können, müssen die beiden Steuer-Eingänge S_0 und S_1 jeweils ein H-Signal haben. Damit liegen die Adressen direkt an den Y-Ausgängen. Die beiden Steuer-Eingänge S_0 und S_1 erhalten ihre Signale von dem Multiplexer-Baustein 25157. Gesteuert wird dieser Baustein durch den LADEN/RECHNEN-Schalter. Der Schalter steuert direkt den S-Eingang an. Bei einem H-Signal liegen die B-Eingänge an den Y-Ausgängen des Bausteines. Da der 2B- und der 3B-Eingang mit $+U_b$ verbunden ist, haben die Ausgänge 2Y und 3Y ein H-Signal und die Adressen-Schalter des Bausteines 2909 sind direkt mit den Y-Ausgängen verbunden.
Mit dem Schalter LADEN/RECHNEN bestimmen wir den Lade- oder den Rechen-Betrieb des Mikrocomputer-Systems. Es gilt für den Schalter folgende Funktionstabelle:

L = Rechnen,
H = Laden.

Im Rechen-Betrieb liegen die Ausgänge Q_0 bis Q_3 an dem Mikroprogramm-Sequenzer. Der Ausgang Q_0 ist direkt verbunden und die Ausgänge Q_1, Q_2 und Q_3 über dem Multiplexer-Baustein U21. Die anderen Ausgänge des Festwertspeichers sind mit den OR_0-, OR_1- und OR_2-Eingängen des Mikroprogramm-Sequenzers verbunden. Erhalten diese Eingänge ein Signal, so können sie die Ausgänge des Bausteines 2909 direkt beeinflussen.
Der Festwertspeicher-Baustein U27 decodiert die Befehle aus dem RAM U7. Dieses Decodieren des Befehls ist notwendig, damit der Mikroprozessor arbeiten kann. Wir geben die nächsten Befehle für den Mikroprozessor in das RAM U7 nach einem Codewort-Schlüssel ein. Der Festwertspeicher erhält bei einem Aufruf des RAMs U7 die einzelnen Codewörter und decodiert sie so, daß der Mikroprozessor arbeiten kann. Es ergibt sich folgende Definition für die einzelnen Codewörter:



2.5 Verschachtelung des Mikroprogramm-Sequenzers

Code-wort	CCE	MUX-Baustein	Stapel-Speicher	OR-Eingänge	Status-Register
0	0	Reg	Inhibit	L	Inhibit
	1	PC			
1	0	Reg	Inhibit	L	Enable
	1				
2	0	PC	Inhibit	L	Enable
	1				
3	0	D	Inhibit	L	Enable
	1				
4	0	Reg	Enable & PUSH	L	Inhibit
	1	PC	Inhibit		
5	0	Reg	Enable & PUSH	L	Enable
	1				
6	0	Stack	Enable & POP	L	Enable
	1				
7	0	Stack	Inhibit	L	Enable
	1				

Code-wort	CCE	MUX-Baustein	Stapel-Speicher	OR-Eingänge	Status-Register
8	0	Stack	Inhibit	L	Inhibit
	1	PC	Enable & POP		
9	0	PC	Enable & PUSH	L	Enable
	1				
10	0	PC	Enable & POP	L	Enable
	1				
11	0	Stack	Inhibit	L	Inhibit
	1	PC	Enable & POP		
12	0	PC	Inhibit	L	Inhibit
	1	Reg			
13	0	PC	Inhibit	L	Inhibit
	1	Reg			
14	0	PC	Inhibit	L	Inhibit
	1	Reg			
15	0	PC	Inhibit	L	Inhibit
	1	Reg			

Aus dieser Tabelle wurde nun die Programmierung für den Festwertspeicher 29751 aufgestellt. Es ergibt sich folgende Programmiertabelle:

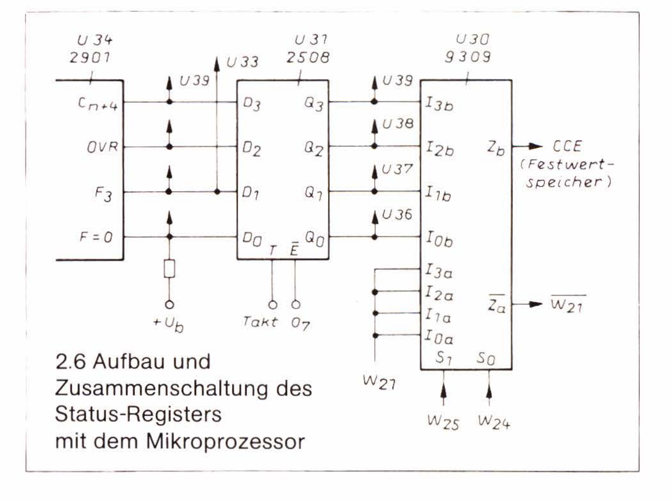
Codewort	Adressen					Ausgänge							
	A_4	A_3	A_2	A_1	A_0	Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
0	L	L	L	L	L	H	L	L	L	H	L	H	L
	L	L	L	L	H	H	L	L	L	L	L	H	L
1	L	L	L	H	L	L	L	L	L	H	L	H	L
	L	L	L	H	H	L	L	L	L	H	L	H	L
2	L	L	H	L	L	L	L	L	L	L	L	H	L
	L	L	H	L	H	L	L	L	L	L	L	H	L
3	L	L	H	H	L	L	L	L	L	H	H	H	L
	L	L	H	H	H	L	L	L	L	H	H	H	L
4	L	H	L	L	L	H	L	L	L	H	L	L	H
	L	H	L	L	H	H	L	L	L	L	L	H	L
5	L	H	L	H	L	L	L	L	L	H	L	L	H
	L	H	L	H	H	L	L	L	L	H	L	L	H
6	L	H	H	L	L	L	L	L	L	L	H	L	L
	L	H	H	L	H	L	L	L	L	L	H	L	L
7	L	H	H	H	L	L	L	L	L	L	H	H	L
	L	H	H	H	H	L	L	L	L	L	H	H	L
8	H	L	L	L	L	H	L	L	L	L	H	H	L
	H	L	L	L	H	H	L	L	L	L	L	L	L
9	H	L	L	H	L	L	L	L	L	L	L	L	H
	H	L	L	H	H	L	L	L	L	L	L	L	H
10	H	L	H	L	L	L	L	L	L	L	L	L	L
	H	L	H	L	H	L	L	L	L	L	L	L	L
11	H	L	H	H	L	H	L	L	L	L	H	H	L
	H	L	H	H	H	H	L	L	L	L	L	L	L
12	H	H	L	L	L	H	L	L	L	L	L	H	L
	H	H	L	L	H	H	L	L	L	H	L	H	L
13	H	H	L	H	L	H	L	L	L	L	L	H	L
	H	H	L	H	H	H	L	L	L	H	L	H	L
14	H	H	H	L	L	H	L	L	L	L	L	H	L
	H	H	H	L	H	H	L	L	L	H	L	H	L
15	H	H	H	H	L	H	L	L	L	L	L	H	L
	H	H	H	H	H	H	L	L	L	H	L	H	L

Der Ausgang Q_0 des Festwertspeichers ist direkt mit dem PUP-Eingang des Mikroprogramm-Sequenzers 2909 verbunden. Mit einem L-Signal haben wir einen POP-Betrieb und mit einem H-Signal den PUSH-Betrieb. Die gespeicherten Daten des Kellerspeichers werden aber nur dann auf die Ausgänge des Sequenzers geschaltet, wenn der S_1-Eingang ein H-Signal und der S_0-Eingang ein L-Signal hat, d. h. wenn der Q_3- und Q_2-Ausgang des Festwertspeichers diese Signale aufweist. Der $\overline{FE}$-Eingang des Bausteines 2909 ist mit dem Q_1-Ausgang verbunden.

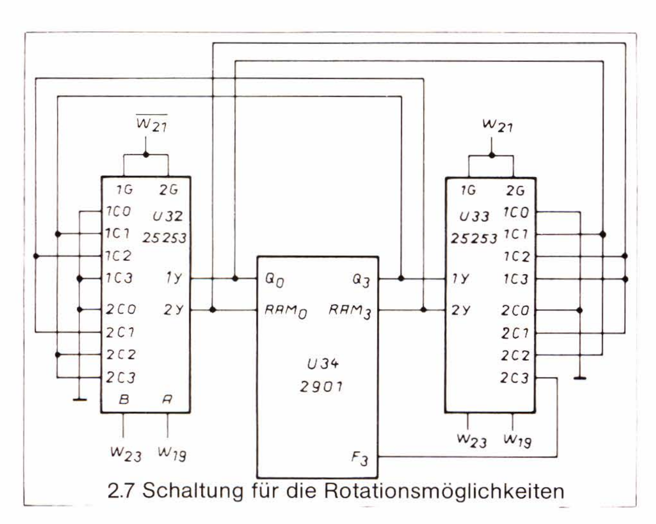
Die Q_4-, Q_5- und Q_6-Ausgänge steuern die OR-Eingänge des Mikroprogramm-Sequenzers an. In der Programmiertabelle haben wir aber keine Programmierung, d. h. die Ausgänge des Festwertspeichers beeinflussen den Sequenzer nicht. Wechselt man den Festwertspeicher gegen einen anderen mit einer anderen Programmierung aus, so verändern sich die einzelnen Funktionen des Mikrocomputer-Systems.

Der Q_7-Ausgang steuert das Status-Register U31 an. Das Status-Register speichert die vier Funktionsausgänge C_{n+4}, OVR, F_3 und $F = 0$ zwischen. Damit können wir den Status des Mikroprozessors überwachen. Mit der Speicherung des Status kann der Mikroprozessor eine Entscheidung treffen. Wenn wir uns das Bild 2.6 betrachten, sehen wir den Speicher-Baustein U31 nach den vier Funktionsausgängen des Mikroprozessors. Diese vier Funktionen werden gespeichert und liegen über dem Multiplexer-Baustein U30 an dem CCE-Eingang des Festwertspeichers an. Wir erhalten hiermit eine Status-Schleife, die es dem Mikrocomputer-System ermöglicht, Entscheidungen zu treffen.

In Bild 2.6 ist der Aufbau des Status-Registers in Verbindung mit dem Mikroprozessor gezeigt. Die einzelnen Funktionen werden in dem Speicher U 31 zwischengespeichert und auf den Multiplexer U30 geschaltet. Die Zwischenspeicherung erfolgt nur, wenn der Eingang $\overline{E}$ durch den Festwertspeicher angesteuert und ein positiver Taktsprung durchgeführt wird. Über den Multiplexer U30 wird der Ausgang CCE erzeugt. Welcher



2.6 Aufbau und Zusammenschaltung des Status-Registers mit dem Mikroprozessor

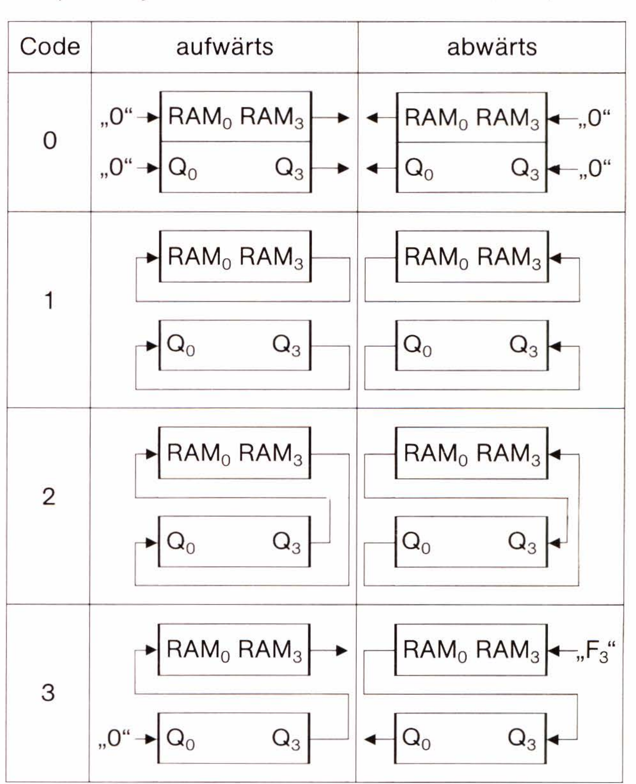


2.7 Schaltung für die Rotationsmöglichkeiten

der vier Status-Eingänge auf den Ausgang geschaltet werden soll, bestimmen die beiden Eingänge S_1 und S_0. Die beiden Eingänge beziehen ihre Signale aus dem RAM U7. Es ergibt sich folgende Funktionstabelle:

S_1	S_0	Funktion
L	L	$F = 0$
L	H	F_3
H	L	OVR
H	H	C_{n+4}

Bild 2.7 zeigt die Rotationsmöglichkeiten des Mikrocomputer-Systems. Hier die Schiebefunktionen:



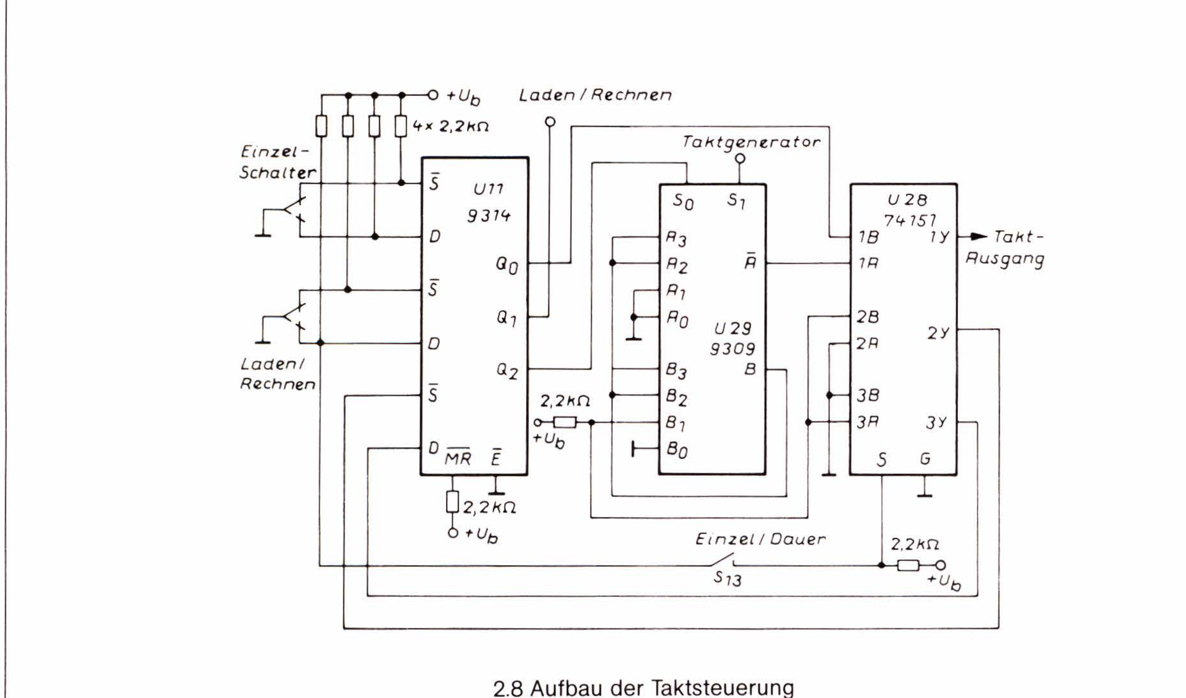
Ist die betreffende Funktion nicht erfüllt, so hat der CCE-Ausgang ein L-Signal. Bei einer erfüllten Funktion hat der Ausgang ein H-Signal. Mit diesem CCE-Signal wird der Eingang des Festwertspeichers angesteuert und damit innerhalb eines Codewortes variiert.
Der andere Teil des Bausteines U30 wird als NICHT-Gatter verwendet. Damit wird ein Baustein gespart.

Das Codewort wird mit dem MUX_1- und MUX_0-Eingang in das RAM U8 und U6 eingegeben. Dadurch ergibt sich folgende Funktionstabelle:

Code-wort	MUX_1	MUX_0	aufwärts	abwärts
0	L	L	Null einschieben	Null einschieben
1	L	H	Rotieren	Rotieren
2	H	L	Verdoppeln	Verdoppeln
3	H	H	$0 \rightarrow Q_0$ (verdoppeln)	$F_3 \rightarrow RAM_3$ (verdoppeln)

Die Betriebsarten für die einzelnen Schiebefunktionen werden durch die beiden Bausteine U32 und U33 erzeugt. Die Ansteuerung der beiden Bausteine wird durch die RAM-Ausgänge W_{23} und W_{19} vorgenommen, die wir entsprechend unseres Programmes programmieren können.

In Bild 2.8 ist die Taktsteuerung für das Mikrocomputer-System gezeigt. Der Taktgenerator ist nicht eingezeichnet. In dem Handbuch der TTL-Technik Teil 3, TOPP-Nummer 117 ist auf der Seite 52 ein einfacher, aber sehr konstanter und genauer Taktgenerator gezeigt. Hierzu wird der TTL-Baustein 7413 verwendet, der mit den Mikrocomputer-System-Bausteinen in der Betriebsspannung und in den anderen technischen Daten weitgehend identisch ist. Der Taktgenerator soll eine Frequenz zwischen 1 kHz und 1 MHz haben, je nach Anwendungsfall.
Dieser Taktgenerator liegt an dem Eingang S_1 des Bausteines U29. Mit dem Schalter „Einzel/Dauer“ bestimmen wir, ob die Frequenz des Taktgenerators oder des „Einzel/Schalters“ an dem Takt-Ausgang anliegt. Durch den „Einzel-Schalter“ können wir Einzeltaktimpulse erzeugen. Diese sind aber nur wirksam, wenn der Schalter „Einzel/Dauer“ auf „Einzel“-Betrieb steht. Mit dem Schalter „LADEN/RECHNEN“ bestimmen wir, ob Daten in das Mikrocomputer-System eingeschrieben werden sollen oder ob das System arbeiten soll.
Das Mikrocomputer-System kann mit der Frequenz des Taktgenerators oder mit Einzeltakt arbeiten. Bei dem Betrieb durch einen einstellbaren Taktgenerator sehen wir die Schnelligkeit des Rechenvorganges. Erzeugen wir den Einzeltakt mit dem Schalter, so können wir jeden einzelnen Rechenschritt des Systems nachvollziehen.

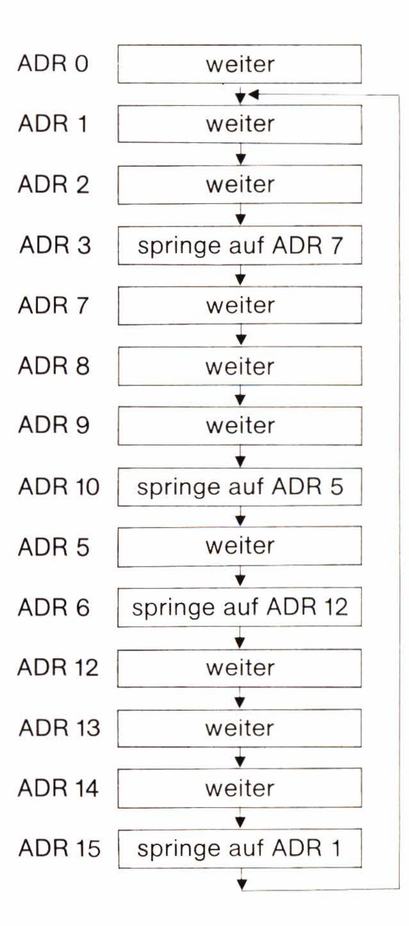


2.8 Aufbau der Taktsteuerung

3. Programmierung

Programmierungsbeispiel 1
Adressierung und unbedingte Sprünge

In der Programmiertabelle 1 ist der Ablauf des Programmes gezeigt. Es ergibt sich nebenstehende Funktionstabelle.
Mit dem Befehl „weiter“ erhöht das Mikrocomputer-System seine Adresse immer um eine Eins. Daher wird der Befehl „weiter“ auch als „ADR+1“ bezeichnet. Durch einen unbedingten Sprungbefehl springt der Mikroprozessor auf die Adresse, die wir in der Spalte 7 festlegen. Dabei kann der Mikroprozessor einen Sprung nach oben oder nach unten durchführen.
Bei dem Beispiel 1 beginnt der Mikroprozessor bei der Adresse ADR 0. Danach erfolgt pro Taktimpuls eine Erhöhung der Adresse. Bei der ADR 3 springt er auf die ADR 7 und setzt dort sein Programm fort. Von der ADR 10 springt er auf die ADR 5 und setzt dort sein Programm weiter. Das Programm würde auf der ADR 15 enden, aber hier wurde ein Sprung-Befehl programmiert. Der Mikroprozessor springt daher auf ADR 1 und beginnt sein Programm von neuem. Wir haben durch diesen Sprung eine Schleife, die der Mikroprozessor aber nicht verlassen kann, da keine Entscheidung innerhalb der Schleife getroffen wird.



| RAM & MUX | 7 | | | | 6 | | | | 5 | | | | 4 | | | | 3 | | | | 2 | | | | 1 | | | | 0 | | | | Bemerkungen | |
| | Zweig-Adresse | | | | Befehl | | | | | Lade-Funkt. | | | | Regist. ALU | | | | ALU | | | „A“ | | | | „B“ | | | | Daten | | | | nächste Adresse | Daten |
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0					0	0	1	0																										
1					0	0	1	0																										
2					0	0	1	0																										
3	0	1	1	1	0	0	0	1																									7	
4																																		
5					0	0	1	0																										
6	1	1	0	0	0	0	0	1																									12	
7					0	0	1	0																										
8					0	0	1	0																										
9					0	0	1	0																										
10	0	1	0	1	0	0	0	1																									5	
11																																		
12					0	0	1	0																										
13					0	0	1	0																										
14					0	0	1	0																										
15	0	0	0	1	0	0	0	1																									1	

Programmiertabelle 1 Funktion: Adressierung und unbedingte Sprünge

Programmierungsbeispiel 2
Lade das interne RAM B des Mikroprozessors

In Bild 1.1.2 ist die Innenschaltung des Mikroprozessors 2901 gezeigt. In diesem Programmierbeispiel soll nun das Laden des internen RAMs B erklärt werden.
Das Einschreiben der Informationen (Daten) erfolgt über die direkten Daten-Eingänge des Mikroprozessors. Die Daten gelangen über den Daten-Selektor, die ALU und den Ausgangsdaten-Selektor vor den internen RAMs auf das RAM B. Hierzu sind in der Zeile 0 (Adresse 0) folgende Befehle notwendig:

Spalte 0: Hier haben wir eine Zahl 8 gespeichert, die in das RAM B eingeschrieben werden soll.

Spalte 1: Hier steht die Adresse des RAM B, auf der die Zahl 8 eingeschrieben werden soll.

Spalte 3: Hier steht der Befehl für die ALU. In dem Programmierbeispiel wurde eine ODER-Verknüpfung gewählt. Diese Verknüpfung ist bei Übertragungsfunktionen innerhalb der ALU zu empfehlen.

Spalte 4: Hier bestimmen wir, welche Eingänge auf die ALU geschaltet werden. Bei dem Programmierbeispiel ist der Eingang D, also der direkte Daten-Eingang mit der ALU verbunden. Die S-Eingänge der ALU haben ein 0-Wort. Deshalb ist es zweckmäßig einer ODER-Verknüpfung als Befehl für die ALU zu wählen.

Spalte 5: Auf der Adresse 0 wird in dieser Spalte die Ladefunktion festgelegt, d. h. die F-Ausgänge der ALU sind mit den Eingängen des RAM B zu verbinden.
Wird auf das Mikrocomputer-System ein Taktsignal gegeben, so erfolgt die Speicherung der Zahl 8 auf der Adresse 0 in dem RAM B.
Auf der ADR 1 in unserer Programmiertabelle 2 können wir den gespeicherten Inhalt des RAM B und der Adresse 0 zerstörungsfrei auslesen. Hierzu verbindet der Befehl in der Spalte 4 den Ausgang des RAM B mit der ALU. In der ALU wird wieder eine ODER-Verknüpfung durchgeführt und in der Spalte 5 haben wir keine Ladefunktion. Die Zahl 8 steht uns an den Y-Ausgängen des Mikroprozessors zur Verfügung.
Bei der ADR 2 schreiben wir in der Programmiertabelle eine 3 auf den Speicherplatz 1 in das RAM B ein. Danach wird der Speicherplatz zerstörungsfrei ausgelesen.
Bei der ADR 4 schreiben wir in der Programmiertabelle eine 12 auf den Speicherplatz 2 in das RAM B ein. Danach lesen wir zerstörungsfrei den Speicherplatz aus.
In der ADR 6 wird eine 7 eingeschrieben und ausgelesen.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A"				„B"				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	1	1		1	1	1		0	1	1					0	0	0	0	1	0	0	0		8
1										0	0	1		0	1	1		0	1	1					0	0	0	0						
2										0	1	1		1	1	1		0	1	1					0	0	0	1	0	0	1	1		3
3										0	0	1		0	1	1		0	1	1					0	0	0	1						
4										0	1	1		1	1	1		0	1	1					0	0	1	0	1	1	0	0		12
5										0	0	1		0	1	1		0	1	1					0	0	1	0						
6										0	1	1		1	1	1		0	1	1					0	0	1	1	0	1	1	1		7
7										0	0	1		0	1	1		0	1	1					0	0	1	1						
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

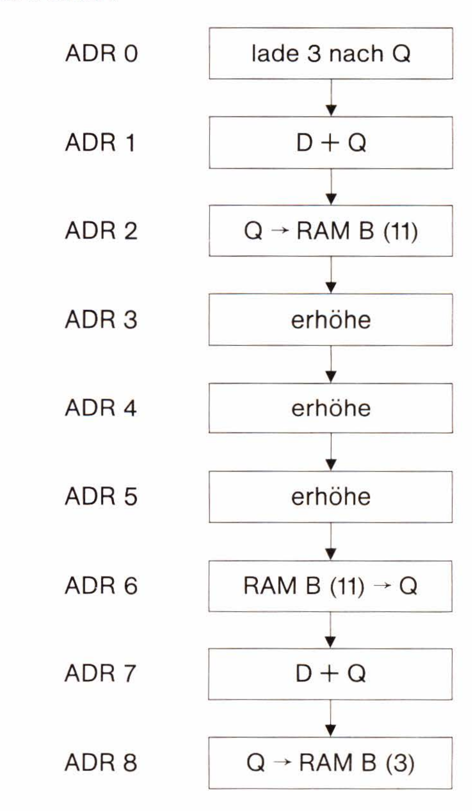
Programmiertabelle 2 Funktion: Lade das interne RAM B des Mikroprozessors

Programmierungsbeispiel 3
Addition über RAM B und Q-Register

In diesem Programm wird die Möglichkeit der Addition über das RAM B und dem Q-Register gezeigt. Auf der Adresse 0 wird die Zahl 3 in das Q-Register eingeschrieben. Auf der Adresse 1 erfolgt eine Addition zwischen dem Q-Register und der Zahl 6. Die Zahl 6 wird aus dem Daten-RAM entnommen und über den direkten Daten-Eingang des Mikroprozessors auf die ALU geschaltet. Das Ergebnis der Addition wird in dem RAM B auf dem Speicherplatz 11 eingeschrieben.
Auf der Adresse 6 beginnen wir mit einer weiteren Addition. Zuerst wird der Speicherplatz 11 des RAM B in das Q-Register eingeschrieben. Danach kann der Inhalt des Q-Registers mit den Daten des direkten Einganges addiert werden. Das Ergebnis der Addition wird in dem Q-Register zwischengespeichert und anschließend auf dem Speicherplatz 3 des RAM B eingeschrieben.

Nach der Adresse 8 ist das Additionsprogramm abgeschlossen. Soll das Programm wiederholt werden, so muß nach der Adresse 8 ein unbedingter Sprungbefehl eingegeben werden.

Von der Programmiertabelle 3 kann ein Flußdiagramm erstellt werden.

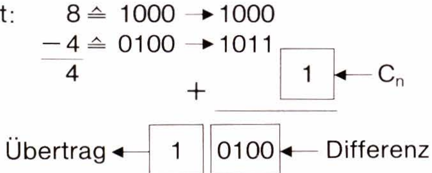


RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									0	0	1	1		3
1										0	0	0		1	1	0		0	0	0									0	1	1	0		6
2										0	1	1		0	1	0		0	1	1					1	0	1	1					RAM B (11)	
3					0	0	1	0																										
4					0	0	1	0																										
5					0	0	1	0																										
6										0	0	0		0	1	1		0	1	1					1	0	1	1					RAM B (11)	
7										0	0	0		1	1	0		0	0	0									0	1	0	0		4
8										0	1	1		0	1	0		0	1	1					0	0	1	1					RAM B (3)	
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 3 Funktion: Addition über RAM B und Q-Register

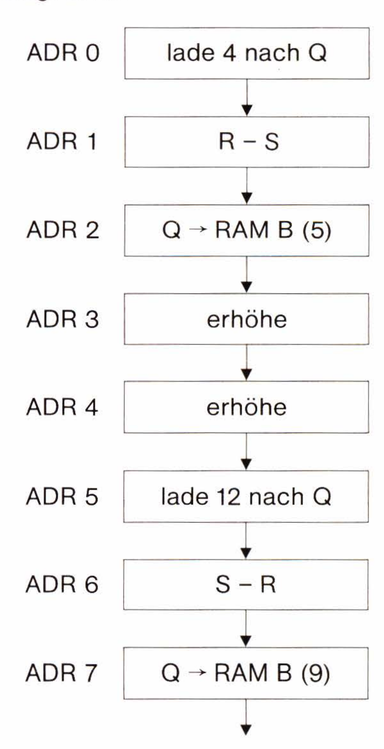
Programmierungsbeispiel 4
Subtraktions-Programme

Der Mikroprozessor 2901 arbeitet nach dem 2^n-Komplement (10er-Komplement). Daher muß ein Übertrag C_n programmiert werden, wie das nachfolgende Beispiel zeigt:



Bei der ersten Subtraktion befindet sich der Subtrahend in dem Q-Register und wird von dem Minuend subtrahiert. Die Differenz der Rechenoperation steht anschließend im Q-Register. Damit wird der Subtrahend gelöscht. Von dem Q-Register wird die Differenz in das RAM B auf den Speicherplatz Sp 5 transportiert.
Bei der zweiten Subtraktion befindet sich der Minuend in dem Q-Register und der Subtrahend wird davon subtrahiert. Die Differenz wird in dem Q-Register gespeichert, wobei der Minuend gelöscht wird. Von dem Q-Register wird die Differenz in das RAM B auf den Speicherplatz Sp 9 transportiert.
Beide Subtraktionsmöglichkeiten sind möglich. Wir müssen beim Programmieren nur auf den Übertrag C_n achten. Vergessen wir den Übertrag, so entsteht eine Differenz minus Eins.

Der Mikroprozessor kann zwei Subtraktionen durchführen, S − R und R − S. In diesem Programmierbeispiel sind beide Möglichkeiten gezeigt. Wir erhalten folgendes Flußdiagramm:

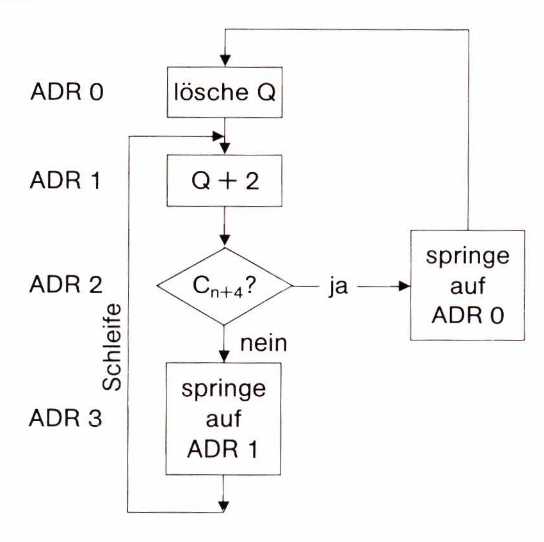


RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A"				„B"				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									0	1	0	0		4
1										0	0	0		1	1	0	1	0	1	0									1	0	0	0		8
2										0	1	1		0	1	0		0	1	1					0	1	0	1					RAM B (5)	
3					0	0	1	0																										
4					0	0	1	0																										
5										0	0	0		1	1	1		0	1	1									1	1	0	0		12
6										0	0	0		1	1	0	1	0	0	1									0	1	1	0		6
7										0	1	1		0	1	0		0	1	1					1	0	0	1					RAM B (9)	
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 4 Funktion: Subtraktions-Programme

Programmierungsbeispiel 5
Schleifenprogramm für eine Addition

Für das Schleifenprogramm ergibt sich folgendes Flußdiagramm:



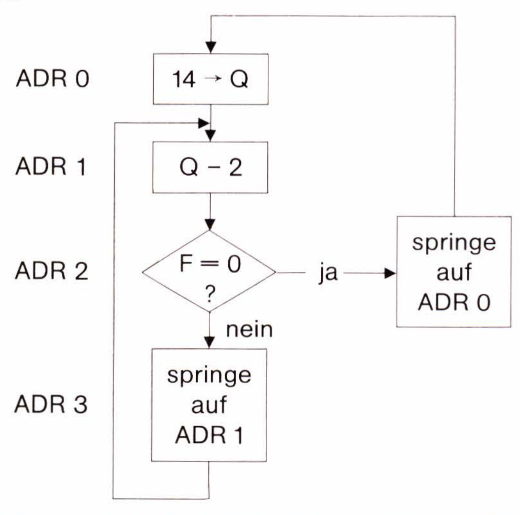
Bei der Adresse ADR 0 wird in das Q-Register die Zahl 0 addiert. Damit wird der gesamte Inhalt des Q-Registers gelöscht.
Bei der Adresse ADR 1 erfolgt die Addition mit einer 2, d. h. mit jeder Schleife wird der Inhalt des Q-Registers um eine 2 erhöht. Dadurch ergeben sich folgende Summen: 2, 4, 6, 8, 10, 12, 14, 0, 2 usw.
Nach der Adresse ADR 2 trifft der Mikroprozessor eine Entscheidung. Diese Entscheidung bezieht sich auf den Übertragsausgang C_{n+4}. Hat der Ausgang ein 0-Signal, erhöht der Mikroprozessor seine Adresse. Von dieser Adresse springt er auf die Adresse 1 zurück und beginnt mit einer neuen Addition.
Die Addition wird solange fortgesetzt, bis der Übertragsausgang des Mikroprozessors ein 1-Signal hat. Hierbei holt sich der Mikroprozessor aus der 7. Spalte seine Sprungadresse und springt auf ADR 0 zurück. Damit kann der Mikroprozessor seinen Q-Inhalt löschen und beginnt wieder von vorne.
Durch dieses Schleifenprogramm kann der Mikroprozessor sein Programm nur dann verlassen, wenn die Entscheidung positiv ist.
Als weiteres Schleifenprogramm für eine Addition kann statt der Zahl 2 eine 3 eingegeben werden. Dadurch verändert sich die Sequenz des Mikroprozessors.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									0	0	0	0		
1										0	0	0		1	1	0		0	0	0									0	0	1	0		2
2	0	0	0	0	1	1	1	1		0	0	1		1	1	0		0	1	1													ADR 0	
3	0	0	0	1	0	0	0	1																									ADR 1	
4																																		
5																																		
6																																		
7																																		
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 5 Funktion: Schleifenprogramm für eine Addition

Programmierungsbeispiel 6
Schleifenprogramm für eine Subtraktion

Für das Schleifenprogramm ergibt sich folgendes Flußdiagramm:



Auf der Adresse ADR 0 wird der Minuend in das Q-Register eingespeichert. Auf der ADR 1 wird von dem Minuend der Subtrahend subtrahiert.
Bei der Adresse ADR 2 erfolgt eine Entscheidung. Mit einer positiven Antwort springt der Mikroprozessor aus der Schleife und beginnt wieder bei der Adresse ADR 0.

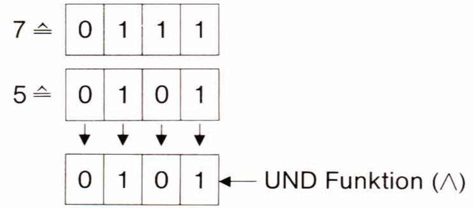
Ist die Entscheidung negativ, erhöht der Mikroprozessor seinen Adressenstand auf ADR 3 und springt dann auf die Adresse ADR 1 zurück.
Der Mikroprozessor befindet sich solange in der Programmschleife, bis die Differenz gleich 0 ist, d. h. wenn der Ausgang F = 0 ein 1-Signal hat. Geben wir einen Minuend mit 13 in die Programmschleife, so kann der Ausgang F = 0 kein 1-Signal erhalten, da $1-2=-1$ ist. Der Mikroprozessor bleibt in der Schleife und setzt das −1 zu 14. Danach beginnt er weiter zu Subtrahieren, bis F = 0 ist.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									1	1	1	0		14
1										0	0	0		1	1	0		0	0	1									0	0	1	0		2
2	0	0	0	0	1	1	0	0		0	0	1		1	1	0		0	1	1													ADR 0	
3	0	0	0	1	0	0	0	1																									ADR 1	
4																																		
5																																		
6																																		
7																																		
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

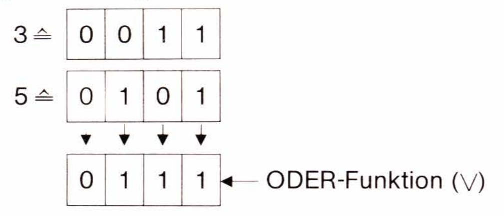
Programmiertabelle 6 Funktion: Schleifenprogramm für eine Subtraktion

Programmierungsbeispiel 7
UND-ODER-Funktionen

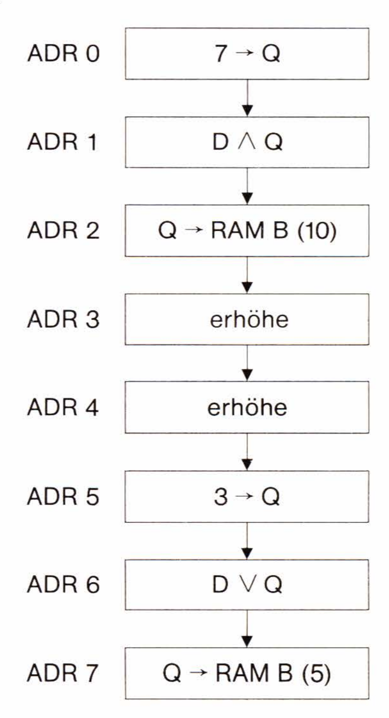
Mit dem Mikroprozessor können logische Funktionen wie UND und ODER ausgeführt werden. Die Verknüpfungen werden immer nur Bitweise vorgenommen.
Auf der Adresse ADR 3 ist eine UND-Verknüpfung zwischen der Zahl 7 und 5. Da die Verknüpfung Bitweise erfolgt, ergibt sich folgende Tabelle:



Die UND-Bedingung ist dann erfüllt, wenn beide Funktionen ein 1-Signal aufweisen.
Auf der Adresse ADR 6 ist eine ODER-Verknüpfung zwischen der Zahl 3 und 5. Da die Verknüpfung Bitweise erfolgt, ergibt sich folgende Tabelle:



Die ODER-Funktion ist dann erfüllt, wenn eine der beiden Funktionen ein 1-Signal hat.
Für das Programmierungsbeispiel ergibt sich folgendes Flußdiagramm:



RAM & MUX	7	6	5		4		3		2	1	0	Bemerkungen	
	Zweig-Adresse	Befehl		Lade-Funkt.		Regist. ALU		ALU	„A“	„B“	Daten	nächste Adresse	Daten
Adr.	BR_3 BR_2 BR_1 BR_0	P_3 P_2 P_1 P_0	MUX_1	I_8 I_7 I_6	MUX_0	I_2 I_1 I_0	C_n	I_5 I_4 I_3	A_3 A_2 A_1 A_0	B_3 B_2 B_1 B_0	D_3 D_2 D_1 D_0		
0				0 0 0		1 1 1		0 1 1			0 1 1 1		7
1				0 0 0		1 1 0		1 0 0			0 1 0 1		5
2				0 1 1		0 1 0		0 1 1		1 0 1 0		RAM B (10)	
3		0 0 1 0											
4		0 0 1 0											
5				0 0 0		1 1 1		0 1 1			0 0 1 1		3
6				0 0 0		1 1 0		0 1 1			0 1 0 1		5
7				0 1 1		0 1 0		0 1 1		0 1 0 1		RAM B (5)	
8													
9													
10													
11													
12													
13													
14													
15													

Programmiertabelle 7 Funktion: UND-ODER-Funktionen

Programmierungsbeispiel 8
Implikations-Funktion

Mit der Implikation wird der R-Eingang der ALU negiert und mit dem S-Eingang konjunktiv (UND-Funktion) verknüpft. Dadurch ergibt sich eine Möglichkeit der Implikations-Funktion und der NICHT-Funktion. Beide Möglichkeiten sind in Tabelle 8 gezeigt.
In dem ersten Beispiel wird die Zahl 3 in das Q-Register eingeschrieben. Die Zahl 5 liegt negiert an und wird konjunktiv verknüpft. Dadurch ergibt sich folgende Tabelle:

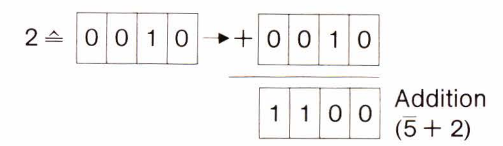
3 ≙ | 0 | 0 | 1 | 1 | → | 0 | 0 | 1 | 1 |

5 ≙ | 0 | 1 | 0 | 1 | → | 1 | 0 | 1 | 0 |

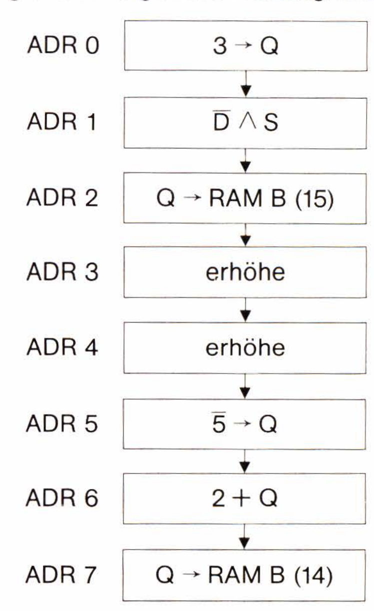
| 0 | 0 | 1 | 0 | Implikation ($3 \wedge \overline{5}$)

Das Produkt der Implikation wird in dem RAM B auf dem Speicherplatz 15 abgespeichert.
In dem zweiten Beispiel liegt die Zahl 5 an dem direkten Dateneingang und wird durch die ALU negiert, da in der Spalte 4 der Befehl 7 für D mit 0 steht. Die Zahl 5 wird direkt negiert in das Q-Register übertragen. Anschließend erfolgt eine Addition zwischen dem Q-Register und dem direkten Dateneingang. Es ergibt sich folgende Tabelle:

5 ≙ | 0 | 1 | 0 | 1 | → | 1 | 0 | 1 | 0 |



Die negierte oder invertierte Zahl 5 ergibt die Zahl 10. Durch die Addition erhalten wir die Summe. Für das Beispiel ergibt sich folgendes Flußdiagramm:



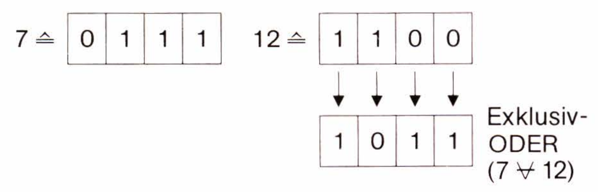
RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A"				„B"				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									0	0	1	1		3
1										0	0	0		1	1	0		1	0	1									0	1	0	1		5
2										0	1	1		0	1	0		0	1	1					1	1	1	1					RAM B (15)	
3					0	0	1	0																										
4					0	0	1	0																										
5										0	0	0		1	1	1		1	0	1									0	1	0	1		5
6										0	0	0		1	1	0		0	0	0									0	0	1	0		2
7										0	1	1		0	1	0		0	1	1					1	1	1	0					RAM B (14)	
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 8 Funktion: Implikations-Funktion

Programmierungsbeispiel 9

Exklusiv-ODER-Funktion

In dem Programmierungsbeispiel sind zwei Möglichkeiten der Exklusiv-ODER-Funktion gezeigt. In der ersten Möglichkeit wird die Zahl 7 in das Q-Register eingeschrieben und dann auf der nächsten Adresse mit einer Zahl 12 verglichen. Wir erhalten folgende Tabelle:



Die Exklusiv-ODER-Funktion wird immer Bitweise durchgeführt. Dabei gilt, sind die beiden Bits ungleich, hat der Ausgang eine 1. Sind die beiden Bits gleich, hat der Ausgang eine 0.

Bei der zweiten Möglichkeit wird eine Zahl 1 negiert in das Q-Register eingeschrieben. Die negierte Zahl wird mit der Zahl 0 verglichen. Es ergibt sich folgende Tabelle:

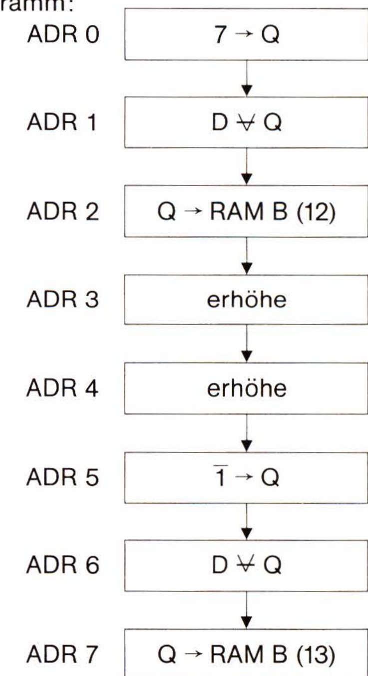
$\overline{1}$ ≙ | 1 | 1 | 1 | 0 | 0 ≙ | 0 | 0 | 0 | 0 |

↓ ↓ ↓ ↓

| 1 | 1 | 1 | 0 | Exklusiv-ODER ($\overline{1}$ ⊻ 0)

Durch die Exklusiv-ODER-Funktion erhalten wir den negierten Wert von der Zahl 1. Diese Funktion der Übertragung kommt in den späteren Programmierungsbeispielen noch vor.

Für das Programmierungsbeispiel ergibt sich folgendes Flußdiagramm:

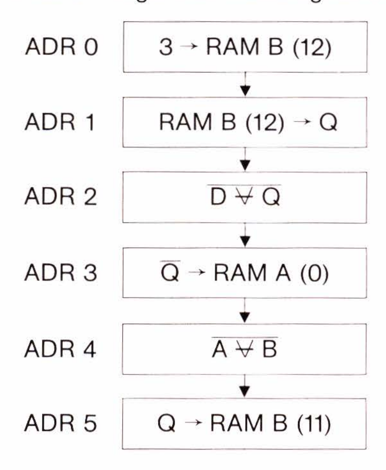


RAM & MUX	7	6	5		4		3		2	1	0	Bemerkungen	
	Zweig-Adresse	Befehl		Lade-Funkt.		Regist. ALU		ALU	„A“	„B“	Daten	nächste Adresse	Daten
Adr.	BR_3 BR_2 BR_1 BR_0	P_3 P_2 P_1 P_0	MUX_1	I_8 I_7 I_6	MUX_0	I_2 I_1 I_0	C_n	I_5 I_4 I_3	A_3 A_2 A_1 A_0	B_3 B_2 B_1 B_0	D_3 D_2 D_1 D_0		
0				0 0 0		1 1 1		0 1 1			0 1 1 1		7
1				0 0 0		1 1 0		1 1 0			1 1 0 0		12
2				0 1 1		0 1 0		0 1 1		1 1 0 0		RAM B (12)	
3		0 0 1 0											
4		0 0 1 0											
5				0 0 0		1 1 1		1 0 1			0 0 0 1		1
6				0 0 0		1 1 0		1 1 0			0 0 0 0		0
7				0 1 1		0 1 0		0 1 1		1 1 0 1		RAM B (13)	
8													
9													
10													
11													
12													
13													
14													
15													

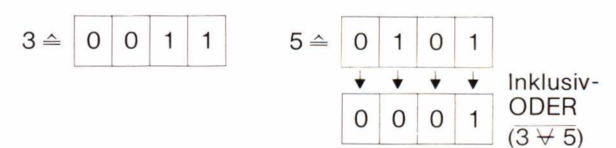
Programmiertabelle 9 Funktion: Exklusiv-ODER-Funktion

Programmierungsbeispiel 10
Inklusiv-ODER-Funktion

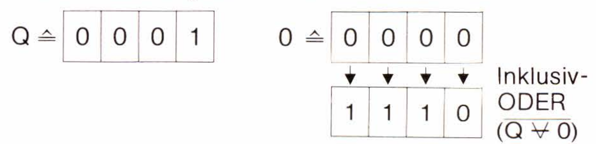
Der Aufbau dieses Programmes ist etwas kompliziert, da Zwischenfunktionen erzeugt und gespeichert werden. Wir erhalten folgendes Flußdiagramm:



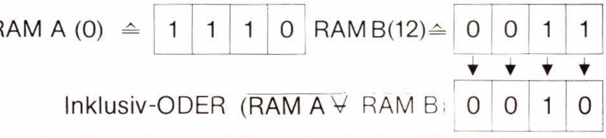
Auf der Adresse ADR 0 schreibt der Mikroprozessor die Zahl 3 in das RAM B auf den Speicherplatz 12 ein. Danach schreibt er zerstörungsfrei die Zahl 3 wieder aus und zwar in das Q-Register.
In der Adresse ADR 2 erfolgt die Inklusiv-ODER-Funktion zwischen dem direkten Dateneingang und dem Q-Register. Wir erhalten folgende Tabelle:



Durch einen Transport-Befehl auf der ADR 2 schreiben wir das Produkt der Inklusiv-ODER-Funktion in das Q-Register ein.
Auf der Adresse ADR 3 führt der Mikroprozessor wieder eine Inklusiv-ODER-Funktion durch, aber mit einem 0-Wort, da die R-Eingänge jeweils ein 0-Signal haben. Wir erhalten folgende Tabelle:



Durch diesen Schritt negieren wir den gesamten Inhalt des Q-Registers. Der negierte Inhalt wird in das RAM A auf den Speicherplatz 0 eingeschrieben.
Auf der Adresse 4 holt sich der Mikroprozessor aus dem Speicherplatz 12 des RAM B die Zahl 3 und vergleicht sie mit dem Inhalt des Speicherplatzes 0 des RAM A. Es ergibt sich folgende Tabelle:



Das Produkt der Funktion wird in dem Q-Register zwischengespeichert und mit der nächsten Adresse in dem RAM B auf dem Speicherplatz 11 abgelegt.

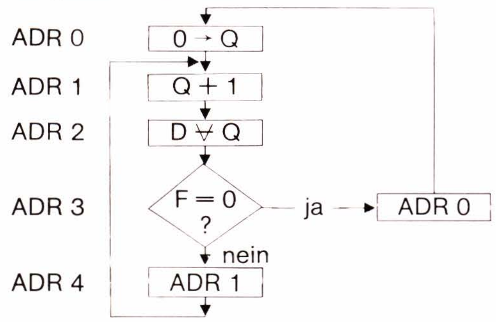
RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	1	1		1	1	1		0	1	1					1	1	0	0	0	0	1	1	RAM B (12)	3
1										0	0	0		0	1	1		0	1	1					1	1	0	0						
2										0	0	0		1	1	0		1	1	1									0	1	0	1		5
3										0	1	0		0	1	0		1	1	1	0	0	0	0										
4										0	0	0		0	0	1		1	1	1	0	0	0	0	1	1	0	0						
5										0	1	1		0	1	0		0	1	1					1	0	1	1					RAM B (11)	
6																																		
7																																		
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 10 Funktion: Inklusiv-ODER-Funktion

Programmierungsbeispiel 11

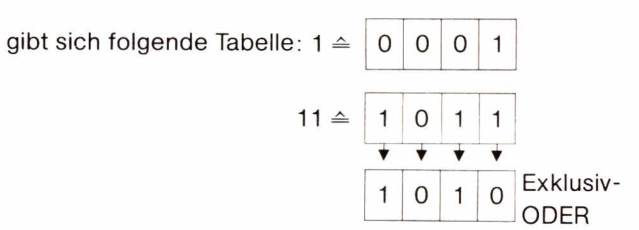
Programmierbarer Vorwärtszähler

Durch das Flußdiagramm kann der Programmablauf vereinfacht dargestellt werden:

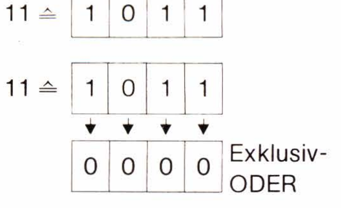


Bei der Adresse ADR 0 wird eine Zahl 0 in das Q-Register eingeschrieben, d. h. der Inhalt des Registers wird gelöscht. Springt der Mikroprozessor aus der Programmschleife auf ADR 0, so löscht er den Inhalt des Registers. Bei der Adresse ADR 1 wird zu dem Inhalt des Q-Registers eine Zahl 1 hinzuaddiert. Bei diesem Programmierungsbeispiel haben wir keine Zahl in dem Daten-RAM stehen, sondern an der ALU einen Übertrag. Durch den Übertrag erhalten wir ein Increment, also ein kontinuierliches Erhöhungsprogramm.

In der Adresse ADR 2 erfolgt die Exklusiv-ODER-Funktion zwischen dem direkten Dateneingang und dem Inhalt des Q-Registers. Nach der ersten Erhöhung ergibt sich folgende Tabelle:



Auf der Adresse ADR 3 wird die Entscheidung getroffen, ob die Exklusiv-ODER-Funktion erfüllt ist oder nicht. Ist die Funktion nicht erfüllt wie in der Tabelle gezeigt, erhöht der Mikroprozessor seine Adresse auf ADR 4 und führt einen unbedingten Sprung auf die ADR 1 aus. Damit durchläuft die Schleife ein Increment und einen nochmaligen Vergleich. Nach der elften Erhöhung ergibt sich folgende Tabelle:



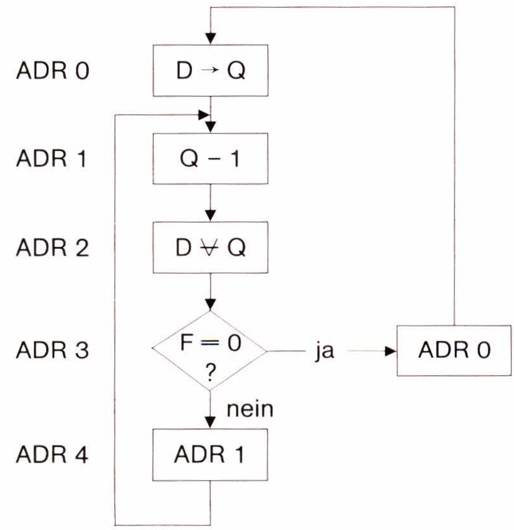
Damit wird die Entscheidung auf der Adresse ADR 3 positiv und der Mikroprozessor springt durch einen bedingten Sprung auf die Adresse ADR 0 zurück. Der Inhalt des Q-Registers wird gelöscht und der Mikroprozessor beginnt wieder von vorne mit seinem Programm.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									0	0	0	0		0
1										0	0	0		0	1	0	1	0	0	0														
2										0	0	1		1	1	0		1	1	0									1	0	1	1		11
3	0	0	0	0	1	1	0	0		0	0	1		0	1	0		0	1	1													ADR 0	
4	0	0	0	1	0	0	0	1		0	0	1		0	1	0		0	1	1													ADR 1	
5																																		
6																																		
7																																		
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 11 Funktion: Programmierbarer Vorwärtszähler

Programmierungsbeispiel 12
Programmierbarer Rückwärtszähler

Für dieses Programm ergibt sich folgendes Flußdiagramm:



Auf der Adresse ADR 0 wird eine Zahl 13 in das Q-Register eingeschrieben. Es kann auch eine andere Zahl sein.
Bei der Adresse ADR 1 wird ein Decrement durchgeführt, d. h. der Inhalt des Q-Registers um eine Eins verringert. Es ergibt sich folgende Tabelle:

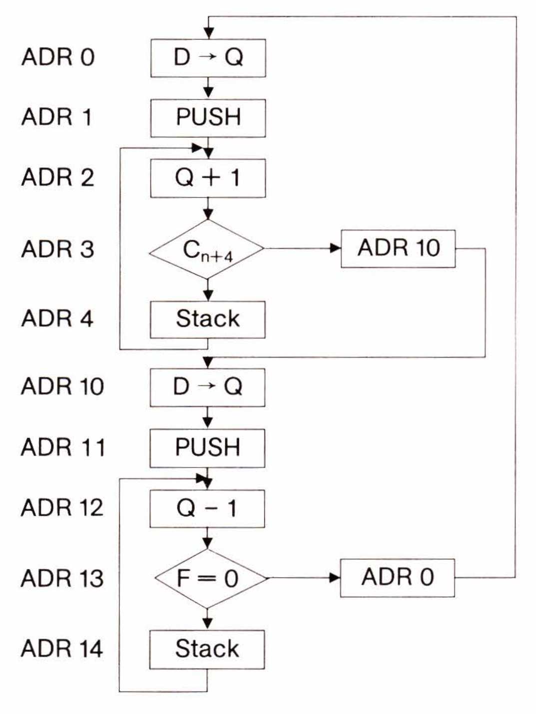
$$\begin{array}{llr} S & & 1101 \triangleq 13 \\ R & + & 1111 \triangleq 15 \\ \hline & & 1\,1100 \triangleq 12 \end{array}$$

Der direkte Dateneingang hat eine Zahl 0. Diese Zahl wird durch den Subtraktions-Befehl invertiert und wir erhalten eine Zahl 15. Die 13 und die 15 werden addiert und es entsteht die Zahl 12 mit einem Übertrag.
Nach der ADR 2 erfolgt die Exklusiv-ODER-Funktion in dem Flußdiagramm. Der Inhalt des Q-Registers wird mit dem direkten Dateneingang verglichen. Danach erfolgt auf der ADR 3 die Entscheidung, ob F = 0 ist. Ist die Entscheidung negativ, erhöht der Mikroprozessor seine Adresse und springt dann auf die ADR 1 zurück und nimmt nochmals das Decrement auf. Ist die Entscheidung positiv, erhält der Mikroprozessor die bedingte Sprungadresse und springt auf ADR 0 zurück.
In dem Programmierungsbeispiel haben wir eine Ausgangsbasis von 13 und eine Vergleichszahl von 5. Nach sieben Schleifen springt der Mikroprozessor auf die Adresse ADR 0 zurück und nimmt das Programm wieder auf. Die Ausgangsbasis für die Programmierung kann einen beliebigen Wert zwischen 0 und 15 aufweisen, ebenso die Vergleichszahl. Daher sind maximal 14 Schleifen möglich.
Die Vergleichszahl kann auch größer als die Ausgangsbasis sein. Dabei springt der Mikroprozessor von dem Q-Register-Inhalt von 0 automatisch auf 15 und decrementiert anschließend wieder herunter.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									1	1	0	1		13
1										0	0	0		0	1	0		0	0	1														
2										0	0	1		1	1	0		1	1	0									0	1	0	1		5
3	0	0	0	0	1	1	0	0		0	0	1		0	1	0		0	1	1													ADR 0	
4	0	0	0	1	0	0	0	1		0	0	1		0	1	0		0	1	1													ADR 1	
5																																		
6																																		
7																																		
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmierungsbeispiel 13
Increment- und Decrement-Programm

Für dieses Programm ergibt sich folgendes Flußdiagramm:



In dem ersten Programmabschnitt wird das Increment, die Erhöhung, in einem Programm gezeigt. Bei der Adresse ADR 2 wird nur ein Übertrag zu dem Inhalt des Q-Registers hinzuaddiert. Es ergibt sich folgende Tabelle:

```
   0000
 + 0000 [1] ← Übertrag
 ------
   0001
```

Mit jeder Schleife erhöht sich der Inhalt des Q-Registers um +1. Ist die Entscheidung positiv, springt der Mikroprozessor auf die Adresse ADR 10.

In dem zweiten Programmabschnitt wird das Decrement, die Verringerung, in einem Programm gezeigt. Bei der Adresse ADR 12 wird das Komplement aus 0 gebildet und dann zum Inhalt des Q-Registers hinzuaddiert. Es ergibt sich folgende Tabelle:

```
               1111
             + 1111
             ------
Übertrag → [1] 1110
```

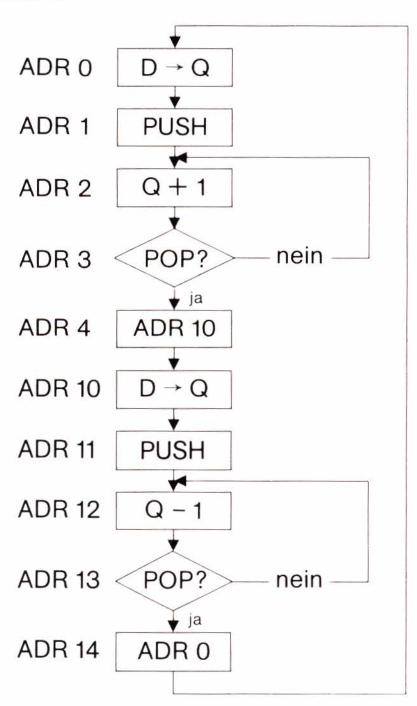
Mit jeder Schleife verringert sich der Inhalt des Q-Registers um –1. Ist die Entscheidung positiv, springt der Mikroprozessor auf die Adresse ADR 0.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									0	0	0	0		0
1					1	0	0	1		0	0	1		0	1	0		0	1	1													PUSH	
2										0	0	0		0	1	0	1	0	0	0														
3	1	0	1	0	1	1	1	1		0	0	1		0	1	0		0	1	1													ADR 10	
4					0	1	1	1		0	0	1		0	1	0		0	1	1													Stapelver.	
5																																		
6																																		
7																																		
8																																		
9																																		
10										0	0	0		1	1	1		0	1	1									1	1	1	1		15
11					1	0	0	1		0	0	1		0	1	0		0	1	1													PUSH	
12										0	0	0		0	1	0		0	0	1														
13	0	0	0	0	1	1	1	1		0	0	1		0	1	0		0	1	1													ADR 0	
14					0	1	1	1		0	0	1		0	1	0		0	1	1													Stapelver.	
15																																		

Programmiertabelle 13 Funktion: Increment- und Decrement-Programm

Programmierungsbeispiel 14
PUSH- und POP-Programm

Für das Programmierungsbeispiel ergibt sich folgendes Flußdiagramm:



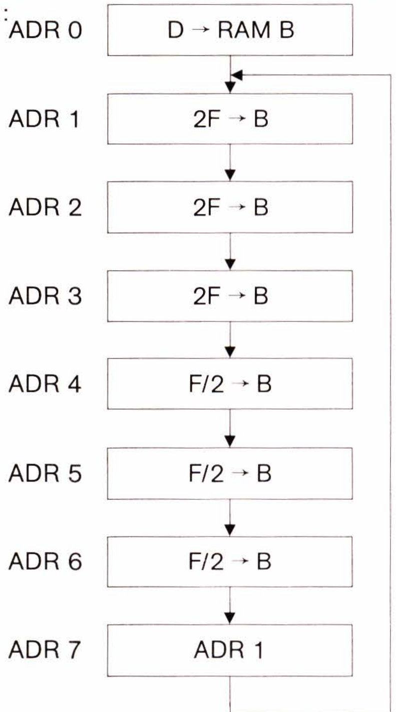
Mit dem Befehl „PUSH" wird die Adresse des Mikroprozessors in den Keller-Speicher (Stapel-Speicher) abgelegt. Danach läuft das Programm weiter. Auf der Adresse ADR 2 im ersten Abschnitt des Flußdiagrammes wird eine 1 zu dem Inhalt des Q-Registers hinzuaddiert. Danach erfolgt auf der ADR 3 ein spezieller Befehl, d. h. Ende der Schleife und POP, wenn C_{n+4} erfüllt ist. Ist die Entscheidung negativ, holt sich der Mikroprozessor aus dem Stapel-Speicher seine Adresse und springt zurück.
Die Schleife wird so lange wiederholt, bis der spezielle Befehl erfüllt ist. Der Mikroprozessor springt auf die Adresse ADR 4 und von dort weiter auf die ADR 10 zu dem nächsten Schleifenprogramm.
Mit dem Befehl „PUSH" wird die Adresse des Mikroprozessors in den Stapel-Speicher abgelegt. Danach haben wir auf der Adresse ADR 12 ein Decrement, also eine Verringerung des Q-Registers. Auf der ADR 13 haben wir wieder einen speziellen Befehl, d. h. Ende der Schleife und POP, wenn F = 0 ist. Ist die Entscheidung negativ, holt sich der Mikroprozessor aus dem Stapel-Speicher seine Adresse und springt zurück.
Die Schleife wird so lange wiederholt, bis der spezielle Befehl der zweiten Programmschleife erfüllt ist. Der Mikroprozessor springt auf die Adresse 14 und von dort aus zu dem ersten Schleifenprogramm zurück.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A"				„B"				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	1	1									0	0	0	0		0
1					1	0	0	1		0	0	1		0	1	0		0	1	1													PUSH	
2										0	0	0		0	1	0	1	0	0	0														
3					1	0	1	1		0	0	1		0	1	0		0	1	1													POP	
4	1	0	1	0	0	0	0	1																									ADR 10	
5																																		
6																																		
7																																		
8																																		
9																																		
10										0	0	0		1	1	1		0	1	1									1	1	1	1		15
11					1	0	0	1		0	0	1		0	1	0		0	1	1													PUSH	
12										0	0	0		0	1	0		0	0	1														
13					1	0	0	0		0	0	1		0	1	0		0	1	1													POP	
14	0	0	0	0	0	0	0	1																									ADR 0	
15																																		

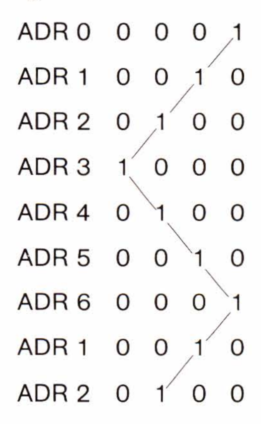
Programmiertabelle 14 Funktion: PUSH-POP-Programm

Programmierungsbeispiel 15
Rechts- und linksschiebendes Programm

Für die Programmiertabelle ergibt sich folgendes Flußdiagramm:



Zuerst wird die Zahl 1 in das RAM B auf den Speicherplatz 0 eingeschrieben. Bei der ADR 1 wird die Zahl um eine Stelle nach links verschoben. Mit jedem Taktimpuls erfolgt eine Verschiebung nach links. Ab der Adresse ADR 4 beginnt der Rechtsbetrieb. Damit ergibt sich folgende Tabelle:



ADR 0	0	0	0	1
ADR 1	0	0	1	0
ADR 2	0	1	0	0
ADR 3	1	0	0	0
ADR 4	0	1	0	0
ADR 5	0	0	1	0
ADR 6	0	0	0	1
ADR 1	0	0	1	0
ADR 2	0	1	0	0

Nach dem Rücksprung auf die Adresse ADR 1 beginnt der Schiebebetrieb wieder von vorne.

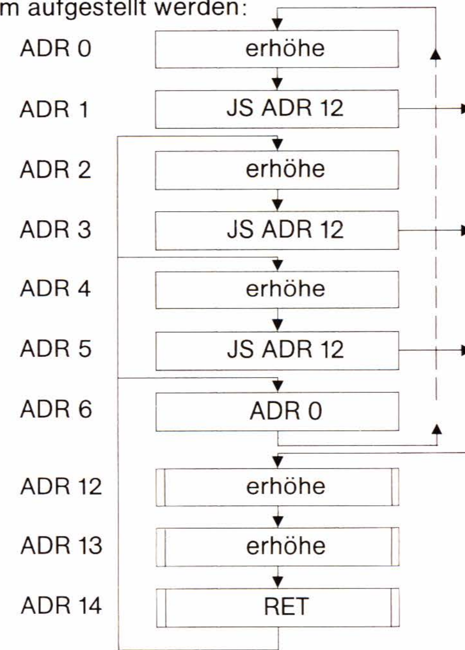
RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	1	1		1	1	1		0	1	1									0	0	0	1		1
1									0	1	1	1	1	0	1	1		0	1	1														
2									0	1	1	1	1	0	1	1		0	1	1														
3									0	1	1	1	1	0	1	1		0	1	1														
4									0	1	0	1	1	0	1	1		0	1	1														
5									0	1	0	1	1	0	1	1		0	1	1														
6									0	1	0	1	1	0	1	1		0	1	1														
7	0	0	0	1	0	0	0	1																									ADR 1	
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 15 Funktion: Rechts- und linksschiebendes Programm

Programmierungsbeispiel 16
Sprung in ein Unterprogramm

Bei dem Hauptprogramm spricht man von einer „Routine" oder „Main-Routine". Bisher haben wir immer mit Routinen gearbeitet. Treten in einem Programm mehrere gleiche Programme auf, so verwenden wir ein Unterprogramm oder eine Subroutine.
In dem Programmierbeispiel haben wir drei Unterprogramm-Sprünge. Mit dem Befehl „JS ADR" springt (jump) der Mikroprozessor auf das Unterprogramm, das mit der Adresse ADR beginnt. Am Ende eines Unterprogrammes haben wir einen Befehl „RET" von return (zurück). Mit diesem Befehl springt der Mikroprozessor in sein Hauptprogramm zurück und fährt dort fort.
Mit dem Befehl „JS ADR wird die Adresse in dem Stapel-Speicher abgelegt. Danach springt der Mikroprozessor auf die Adresse im Zweig-Register. Ist das Unterprogramm abgeschlossen, holt sich der Befehl „RET" die Adresse aus dem Stapel-Speicher und setzt dort sein Programm fort. In dem Beispiel haben wir drei Sprünge in das Unterprogramm. Durch den Befehl „RET" findet der Mikroprozessor seine Adresse in dem Stapel-Speicher. Damit kann das Unterprogramm beliebig oft durch den Mikroprozessor verwendet werden.
In der Praxis legt sich ein System-Entwickler eine Software-Bibliothek zu. Dadurch erhält er im Laufe der Zeit zahlreiche Programme für die einzelnen Probleme.

Bei der Aufstellung eines Programmes verwendet er nach Möglichkeit seine fertigen Programme aus der Bibliothek. Die Programme werden als Unterprogramme in sein Hauptprogramm eingeschaltet. Dadurch sinken die Entwicklungskosten bis auf 80%.
Aus dem Programmierungsbeispiel kann folgendes Flußdiagramm aufgestellt werden:



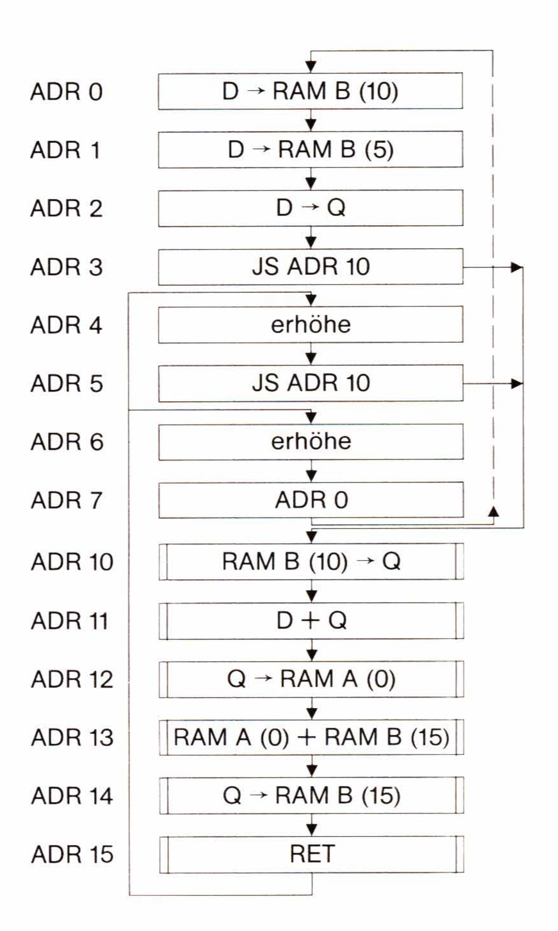
RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0					0	0	1	0																										
1	1	1	0	0	0	1	0	1																										
2					0	0	1	0																										
3	1	1	0	0	0	1	0	1																										
4					0	0	1	0																										
5	1	1	0	0	0	1	0	1																										
6	0	0	0	0	0	0	0	1																										
7																																		
8																																		
9																																		
10																																		
11																																		
12					0	0	1	0																										
13					0	0	1	0																										
14					0	1	1	0																										
15																																		

Programmiertabelle 16 Funktion: Sprung in ein Unterprogramm

Programmierungsbeispiel 17
Einfaches Beispiel für eine Unterprogrammierung

Von der Adresse ADR 3 und ADR 5 erfolgt der Sprung in das Unterprogramm. Das Unterprogramm besteht aus zahlreichen Schritten mit den verschiedenen Rechenoperationen und Schiebebefehlen. Dieses Programm kann z. B. aus einer Software-Bibliothek stammen.
Die Daten für das Unterprogramm werden im Hauptprogramm festgelegt. Mit dem Unterprogramm holen wir uns die einzelnen Daten aus den Speichern und führen damit eine Rechenprogramm durch. Nach dem Abschluß des Unterprogrammes springt der Mikroprozessor automatisch auf die letzte Adresse zurück und nimmt dort sein Hauptprogramm wieder auf.
In dem Unterprogramm können wir je nach Verwendungszweck die Speicherplätze der RAMs anders wählen. Sie sind von dem Hauptprogramm abhängig. Dies gilt auch für die Daten des direkten Einganges.
Die Möglichkeiten eines Unterprogrammes in Verbindung mit einer Software-Bibliothek sind sehr umfangreich und setzt für den Programmierer sehr gute Kenntnisse der Software und der Hardware voraus.

Das nachfolgende Flußdiagramm zeigt den Aufbau des Beispiels:



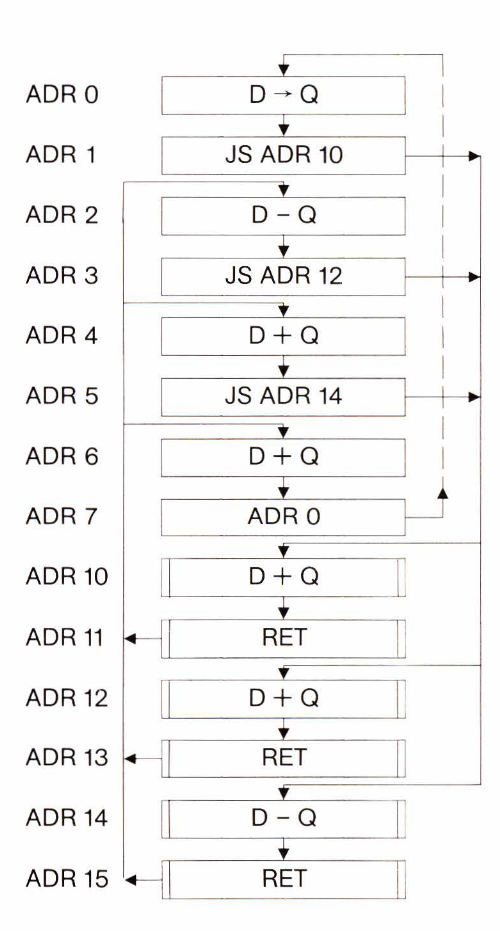
RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A"				„B"				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	1	1		1	1	1		0	1	1					1	0	1	0	0	1	0	0	RAM B (10)	4
1										0	1	1		1	1	1		0	1	1					0	1	0	1	0	0	1	0	RAM B (5)	2
2										0	0	0		1	1	1		0	1	1									0	0	1	0		2
3	1	0	1	0	0	1	0	1																									ADR 10	
4					0	0	1	0																										
5	1	0	1	0	0	1	0	1																									ADR 10	
6					0	0	1	0																										
7	0	0	0	0	0	0	0	1																									ADR 0	
8																																		
9																																		
10										0	0	0		0	1	1		0	1	1					1	0	1	0						
11										0	0	0		1	1	0		0	0	0									0	0	1	1		3
12										0	1	0		0	1	0		0	1	1	0	0	0	0										
13										0	0	0		0	0	1		0	0	0	0	0	0	0	0	1	0	1						
14										0	1	1		0	1	0		0	1	1					1	1	1	1					RAM B (15)	
15					0	1	1	0																										

Programmiertabelle 17 Funktion: Einfaches Beispiel für eine Unterprogrammierung

Programmierungsbeispiel 18
Additions- und Subtraktions-Unterprogramme

Bei diesem Programm springt der Mikroprozessor an drei Stellen seines Hauptprogrammes in ein anderes Unterprogramm. Sicherlich ist das Programierungsbeispiel 18 kein Superprogramm, aber es sollen nur die Möglichkeiten einer Unterprogrammierung und einer Software-Bibliothek gezeigt werden.
Die drei Unterprogramme sind zwei Additionsprogramme und ein Subtraktionsprogramm. Wie das vorherige Beispiel gezeigt hat, können jederzeit auch größere Unterprogramme eingeschaltet werden, was aber von der Speicherkapazität des Systems abhängig ist.
Wichtig bei diesem Programm ist die anschauliche Methode der Unterprogrammierung mit der Adressenspeicherung bei dem Sprungbefehl und dem Rücksprung aus dem Unterprogramm. Hierbei ist deutlich die Funktion des Stapel-Speichers gezeigt, der die letzte Adresse vor dem Unterprogrammsprung festhält und bei einem Rücksprung wieder freigibt.

Für dieses Programmierungsbeispiel ergibt sich folgende Funktionstabelle:

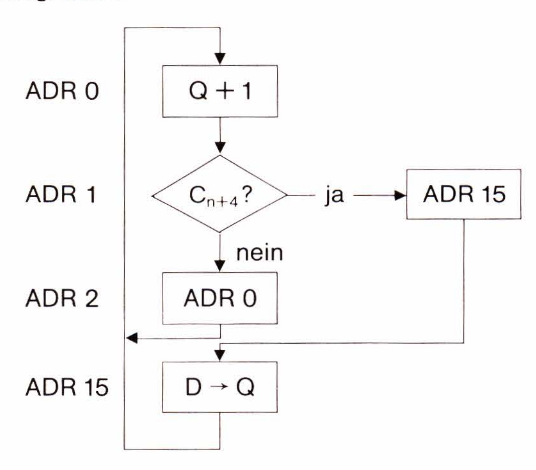


RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0										0	0	0		1	1	1		0	0	0									0	0	1	0		2
1	1	0	1	0	0	1	0	1																									ADR 10	
2										0	0	0		1	1	0		0	0	0									0	0	1	1		3
3	1	1	0	0	0	1	0	1																									ADR 12	
4										0	0	0		1	1	0		0	0	0									1	0	0	0		8
5	1	1	1	0	0	1	0	1																									ADR 14	
6										0	0	0		1	1	0		0	0	0									0	1	1	0		6
7	0	0	0	0	0	0	0	1																									ADR 0	
8																																		
9																																		
10										0	0	0		1	1	0		0	0	0									0	0	1	1		3
11					0	1	1	0																										
12										0	0	0		1	1	0		0	0	0									0	1	0	1		5
13					0	1	1	0																										
14										0	0	0		1	1	0	1	0	0	1									1	1	0	0		12
15					0	1	1	0																										

Programmiertabelle 18 Funktion: Additions- und Subtraktions-Unterprogramme

Programmierungsbeispiel 19
Increment- und Test-Programm

Bei diesem Programm handelt es sich um ein Unterprogramm aus einer Software-Bibliothek. In dem Programmierungsbeispiel wird dieses Unterprogramm getestet und anschließend notiert.
Für dieses Programmierungsbeispiel ergibt sich folgendes Flußdiagramm:



Auf der Adresse 0 wird der Inhalt des Q-Registers durch die ALU incrementiert, also um +1 erhöht. Auf der Adresse 1 erfolgt eine Entscheidung, ob die Bedingung erfüllt ist oder nicht. Ist die Bedingung nicht erfüllt, erhöht der Mikroprozessor seinen Zählerstand und es erfolgt ein unbedingter Sprung auf die Adresse ADR 0. Ist die Bedingung erfüllt, springt der Mikroprozessor auf die ADR 15 und löscht dort den Inhalt des Q-Registers. Danach springt er wieder zurück und nimmt bei der Adresse ADR 0 sein Schleifenprogramm wieder auf.
Auf diese Weise kann ein Programm festgelegt und dokumentiert werden.

RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0					0	0	1	0		0	1	1		0	1	0	1	0	0	0														
1	1	1	1	1	1	1	1	1		0	0	1																					ADR 15	
2	0	0	0	0	0	0	0	1		0	0	1																					ADR 0	
3																																		
4																																		
5																																		
6																																		
7																																		
8																																		
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15	0	0	0	0	0	0	0	1		0	0	0		1	1	1		0	0	0									0	0	0	0		0

Programmiertabelle 19 Funktion: Increment- und Test-Programm

Programmierungsbeispiel 20

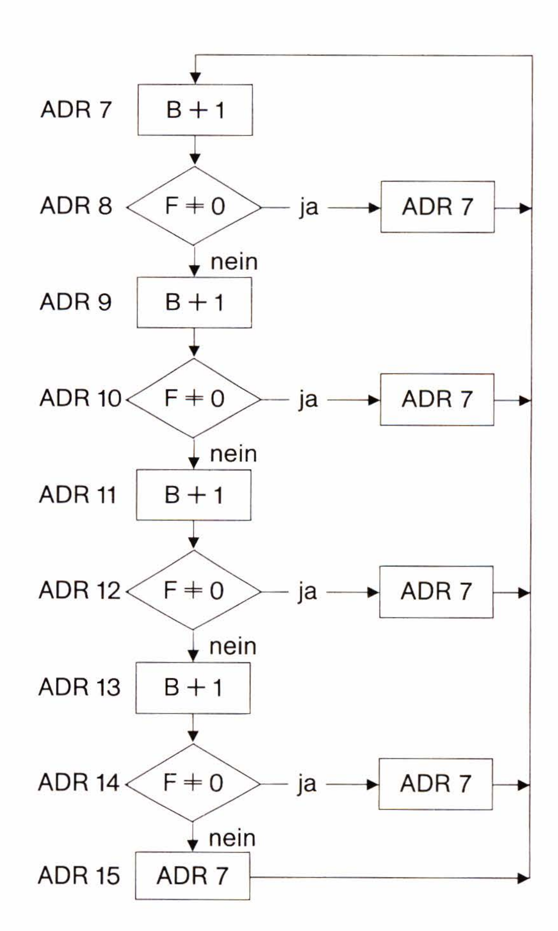
16-Bit-Zähler

Nach jeder Addition wird in diesem Programm eine Entscheidung getroffen. Bei 16 Programmumläufen ist das erste Programmbeispiel abgeschlossen und der Mikroprozessor erhöht seine Adresse auf ADR 9. Hier wird der Inhalt des RAM B Speicherplatz 1 um eine 1 erhöht. Auf der Adresse ADR 10 wird eine Entscheidung getroffen und der Mikroprozessor springt auf ADR 7 zurück. Danach durchläuft er wieder 16 Schleifen und erhöht den Zählerstand auf der ADR 9 um eine weitere 1. Mit dem Programm werden pro Schleife 16 Sprünge benötigt. Da wir vier Schleifen haben, ergeben sich $2^{16} = 65536$ Sprünge. Haben wir einen 1kHz-Takt an dem Mikroprozessor angeschlossen, so dauert es 65,5 Sekunden, bis der Mikroprozessor ADR 15 erreicht.

Nehmen wir bei diesem Programm noch eine Programmschleife hinzu, so erhöht sich das System auf $2^{20} = 1048\,576$ Sprünge. Wird der Mikroprozessor von einem 1 kHz-Takt angesteuert, so dauert es 1048 Sekunden bis der Mikroprozessor ADR 15 erreicht.

Auf dem System können maximal 2^{28} Sprünge programmiert werden, was der Zahl von 268435456 entspricht. Der Mikroprozessor benötigt bei einem 1 kHz-Takt ca. 74 Stunden und 30 Minuten, bis er ADR 15 erreicht.

Für dieses Programm ergibt sich folgendes Flußdiagramm:



RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0																																		
1																																		
2																																		
3																																		
4																																		
5																																		
6																																		
7					0	0	1	0		0	1	1		0	1	1	1	0	0	0					0	0	0	0						
8	0	1	1	1	0	0	0	0		0	0	1																					ADR 7	
9					0	0	1	0		0	1	1		0	1	1	1	0	0	0					0	0	0	1						
10	0	1	1	1	0	0	0	0		0	0	1																					ADR 7	
11					0	0	1	0		0	1	1		0	1	1	1	0	0	0					0	0	1	0						
12	0	1	1	1	0	0	0	0		0	0	1																					ADR 7	
13					0	0	1	0		0	1	1		0	1	1	1	0	0	0					0	0	1	1						
14	0	1	1	1	0	0	0	0		0	0	1																					ADR 7	
15	0	1	1	1	0	0	0	1		0	0	1																					ADR 7	

Programmiertabelle 20 Funktion: 16-Bit-Zähler

Programmierungsbeispiel 21

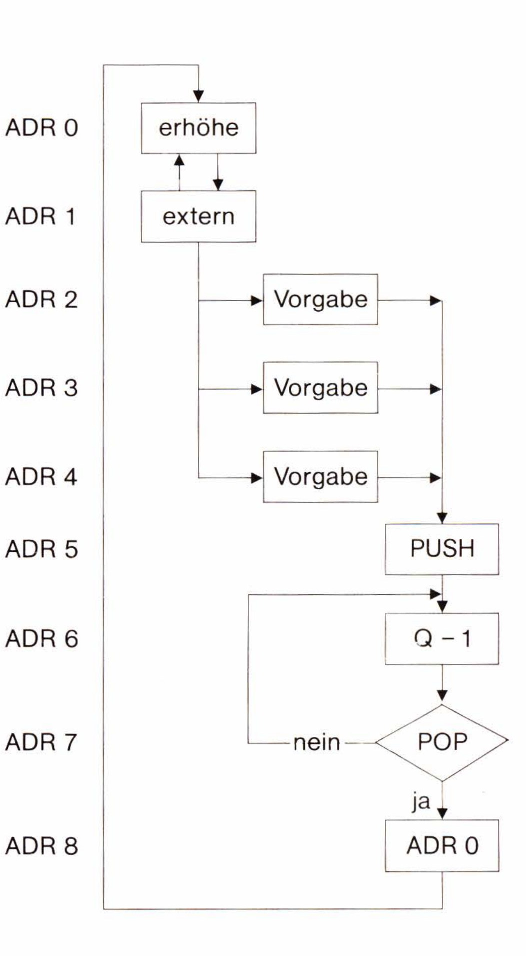
Externe Programmierung

Programmieren wir in der Befehlsspalte die Nummer 3, so können wir den Mikroprozessor extern steuern. Hierzu muß in dem Programm ein Warteschleife vorgesehen sein, d. h. der Mikroprozessor befindet sich in einer Warteschleife und wartet so lange, bis er durch die D-Schalter eine neue Adresse erhält.

Zwischen der Adresse ADR 0 und ADR 1 befindet sich die Warteschleife, d. h. der Mikroprozessor springt zwischen den beiden Adressen so lange hin und her, bis durch die Daten-Schalter eine Adresse eingegeben wird. Bei der Daten-Eingabe kann zwischen der Adresse ADR 2 bis ADR 4 gewählt werden. In den einzelnen Adressen steht die Vorgabe wie folgt: ADR 2: 10
ADR 3: 11
ADR 4: 12

Hat das Q-Register die Vorgabe aufgenommen, springt es durch den unbedingten Sprungbefehl sofort auf die Adresse ADR 5 weiter. Dort wird die Adresse in den Stapel-Speicher eingeschrieben und es beginnt ein Schleifenprogramm mit einem Decrement. Die Schleife ist erst dann beendet, wenn die Bedingung F = 0 erfüllt ist. Der Mikroprozessor erhöht auf die Adresse ADR 0 und springt in die Warteschleife zurück. Danach beginnt der Vorgang wieder von vorne.

Für die Programmiertabelle ergibt sich folgendes Flußdiagramm:



RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0					0	0	1	0		0	0	0		1	1	1		0	1	1									0	0	0	0		0
1					0	0	1	1		0	0	1		0	1	0		0	1	1														
2	0	1	0	1	0	0	0	1		0	0	0		1	1	1		0	0	0									1	0	1	0	ADR 5	10
3	0	1	0	1	0	0	0	1		0	0	0		1	1	1		0	0	0									1	0	1	1	ADR 5	11
4	0	1	0	1	0	0	0	1		0	0	0		1	1	1		0	0	0									1	1	0	0	ADR 5	12
5					1	0	0	1		0	0	1		0	1	0		0	1	1														
6										0	0	0		1	1	0		0	0	1														
7					1	0	0	0		0	0	1		0	1	0		0	1	1														
8	0	0	0	0	0	0	0	1																										
9																																		
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 21 Funktion: Externe Programmierung

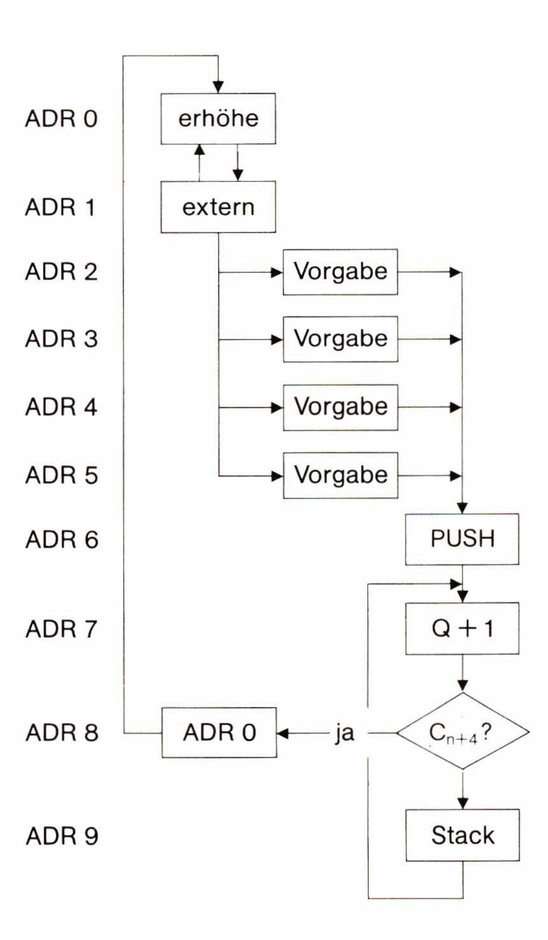
Programmierungsbeispiel 22
Extern programmierbarer Vorwahlzähler

Bei diesem Vorwahlzähler kann zwischen vier Vorgabemöglichkeiten gewählt werden:

ADR 2: 1
ADR 3: 2
ADR 4: 3
ADR 5: 4

Das Ende dieses Programmes tritt bei einem Übertrag C_{n+4} auf. Durch die Vorgabe mittels den Daten-Schaltern können wir zwischen den vier Adressen wählen und so die Vorwahl bestimmen.
Der Ablauf des Programmes ist ähnlich wie beim Programmierungsbeispiel 21. Der Mikroprozessor befindet sich in einer Warteschleife, bis durch die Daten-Schalter die Adresse bestimmt wird. Der Mikroprozessor springt auf die vorgewählte Adresse und holt sich die betreffenden Daten aus dem Schreib-Lese-Speicher. Danach springt er auf die Adresse ADR 6 und nimmt sein Schleifenprogramm auf. Nach Beendigung des Schleifenprogrammes kehrt er wieder auf die Adresse ADR 0 zurück.

Die Vorgabe wird wieder durch die externen Daten-Schalter vorgenommen, wenn sich der Mikroprozessor in der Warteschleife befindet. Es ergibt sich folgendes Programm:



RAM & MUX	7				6				5				4				3				2				1				0				Bemerkungen	
	Zweig-Adresse				Befehl					Lade-Funkt.				Regist. ALU				ALU			„A“				„B“				Daten				nächste Adresse	Daten
Adr.	BR_3	BR_2	BR_1	BR_0	P_3	P_2	P_1	P_0	MUX_1	I_8	I_7	I_6	MUX_0	I_2	I_1	I_0	C_n	I_5	I_4	I_3	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	D_3	D_2	D_1	D_0		
0					0	0	1	0		0	0	0		1	1	1		0	1	1									0	0	0	0		0
1					0	0	1	1		0	0	1		0	1	0		0	1	1														
2	0	1	1	0	0	0	0	1		0	0	0		1	1	1		0	0	0									0	0	0	1	ADR 6	1
3	0	1	1	0	0	0	0	1		0	0	0		1	1	1		0	0	0									0	0	1	0	ADR 6	2
4	0	1	1	0	0	0	0	1		0	0	0		1	1	1		0	0	0									0	0	1	1	ADR 6	3
5	0	1	1	0	0	0	0	1		0	0	0		1	1	1		0	0	0									0	1	0	0	ADR 6	4
6					1	0	0	1		0	0	1		0	1	0		0	1	1														
7										0	0	0		0	0	0	1	0	0	0														
8	0	0	0	0	1	1	1	1		0	0	1		0	1	0		0	1	1													ADR 0	
9					0	1	1	1		0	0	1		0	1	0		0	1	1														
10																																		
11																																		
12																																		
13																																		
14																																		
15																																		

Programmiertabelle 22 Funktion: Extern programmierbarer Vorwahlzähler

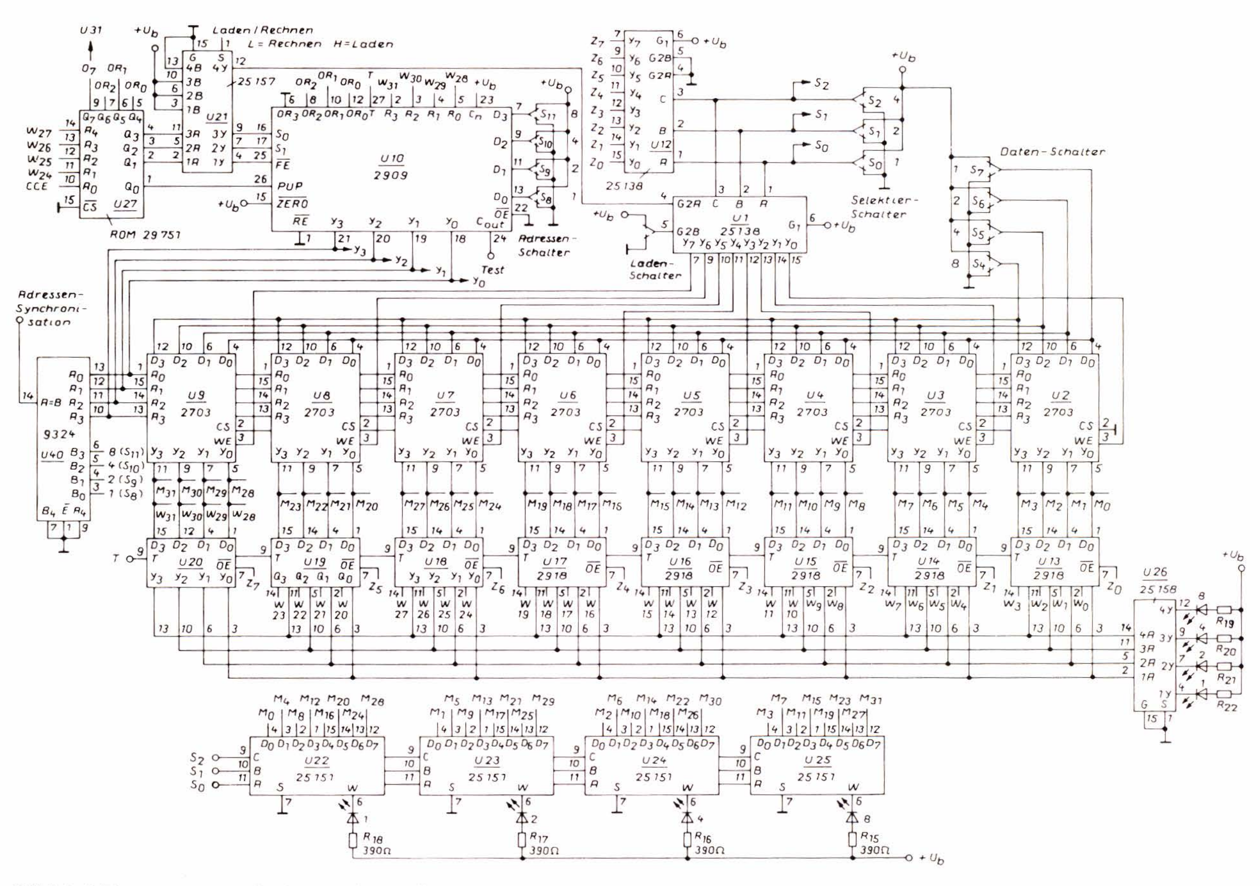


Bild 2.1 Zusammenschaltung des Mikrocomputers

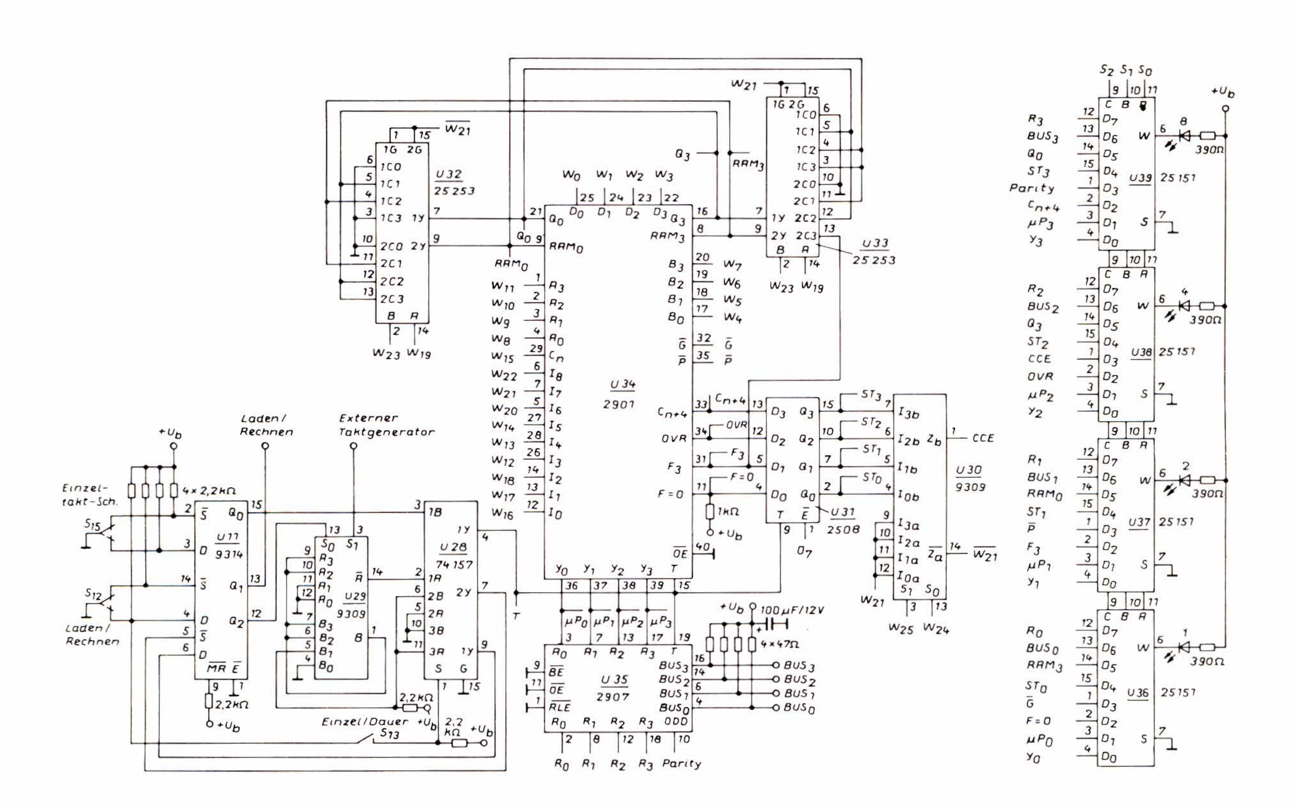
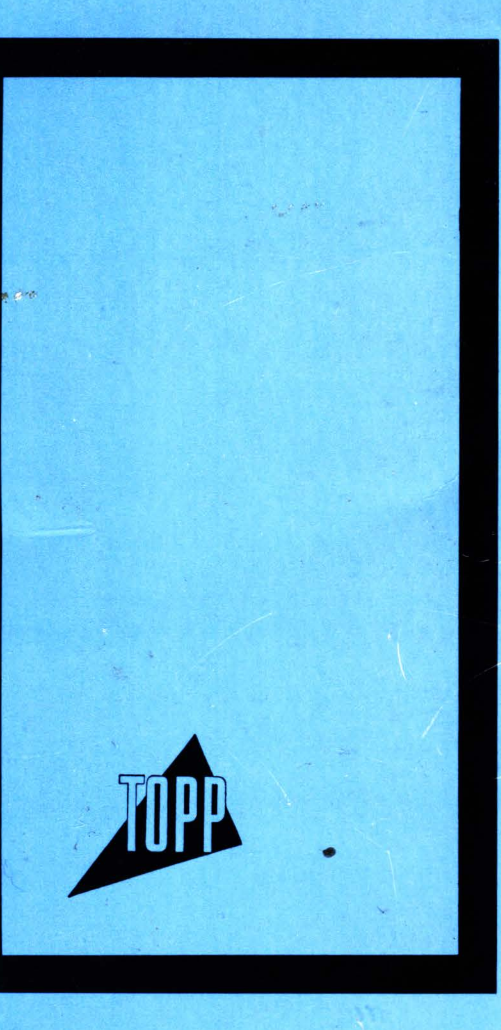


Bild 2.2 Zusammenschaltung des Mikrocomputers



Seit der Elektronica von 1976 in München ist jedem Elektroniker der Begriff des Mikroprozessors bekannt. Bei vielen Elektronikern besteht seitdem der Wunsch nach einem selbstgebauten System, das er in- und auswendig kennt. Dieses System soll aber auch die Möglichkeiten bieten, auf jedes andere Mikroprozessor- oder Mikrocomputersystem schnell umsteigen zu können.

Mit diesem Buch ist selbst ein technisch interessierter Laie in der Lage, sich einen preiswerten Heimcomputer mit einem Mikroprozessor aufzubauen und zu programmieren. Dabei werden Standard-TTL-Bausteine verwendet und ein Mikroprozessor, der von bedeutenden Halbleiterherstellern (AMD, Siemens, NS, MM usw.) produziert wird.

Neben einer ausführlichen Bausteinbeschreibung und Bauanleitung finden Sie 22 Programmierungsbeispiele, die es Ihnen ermöglichen, schnell, einfach und preiswert in die Geheimnisse der Mikroprozessortechnik einzusteigen.

Steckbrief des Verfassers:
Dozent und Applikations-Ingenieur